THE UNIVERSITY
of EDINBURGH

Video object segmentation and applications in temporal alignment and aspect learning

Anestis Papazoglou

Doctor of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2016

Abstract

Modern computer vision has seen recently significant progress in learning visual concepts from examples. This progress has been fuelled by recent models of visual appearance as well as recently collected large-scale datasets of manually annotated still images. Video is a promising alternative, as it inherently contains much richer information compared to still images. For instance, in video we can observe an object move which allows us to differentiate it from its surroundings, or we can observe a smooth transition between different viewpoints of the same object instance. This richness in information allows us to effectively tackle tasks that would otherwise be very difficult if we only considered still images, or even adress tasks that are video-specific.

Our first contribution is a computationally efficient technique for video object segmentation. Our method relies solely on motion in order to rapidly create a rough initial estimate of the foreground object. This rough initial estimate is then refined through an energy formulation to be spatio-temporally smooth. The method is able to handle rapidly moving backgrounds and objects, as well as non-rigid deformations and articulations without having prior knowledge about the objects appearance, size or location. In addition to this class-agnostic method, we present a class-specific method that incorporates additional class-specific appearance cues when the class of the foreground object is known in advance (*e.g.* a video of a car).

For our second contribution, we propose a novel model for temporal video alignment with regard to the viewpoint of the foreground object (*i.e.*, a pair of aligned frames shows the same object viewpoint) Our work relies on our video object segmentation technique to automatically localise the foreground objects and extract appearance measurements solely from them instead of the background. Our model is able to temporally align realistic videos, where events may occur in a different order, or occur only in one of the videos. This is in contrast to previous works that typically assume that the videos show a scripted sequence of events and can simply be aligned by stretching or compressing one of the videos.

As a final contribution, we once again use our video object segmentation technique as a basis for automatic visual aspect discovery from videos of an object class. Compared to previous works, we use a broader definition of an aspect that considers four factors of variation: viewpoint, articulated pose, occlusions and cropping by the image border. We pose the aspect discovery task as a clustering problem and provide an extensive experimental exploration on the benefits of object segmentation for this task.

Acknowledgements

First and foremost, I extend my gratitude to my supervisor, Dr. Vittorio Ferrari for his thoughtful guidance during these past four years. I also am grateful to Dr. Luca Del Pero, whose help was invaluable in my research.

I am very thankful to my family, who were always there for me when I needed it the most. Last, but certainly not least, my most special thanks to my significant other, Galini, whose understanding and encouragement helped me get to this point.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Anestis Papazoglou*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since the first video camera was invented, video has become an integral part of everyday life. While the creation of videos was originally something that required large, expensive equipment the advent of the digital age has significantly changed that. Quality video cameras can now be found in all kinds of consumer electronics, including laptops, tablets, cell phones or even glasses. People nowadays carry a video camera with them wherever they go and take hundreds of hours of videos every year. Thanks to video sharing websites (YouTube, Vimeo) sharing all these videos with friends and strangers alike has become easier than ever. Anyone with a web browser today can have access to hundreds of thousands of videos. Taking into account the videos that are produced professionally (e.g. movies and TV shows) as well as surveillance videos, it is easy to see that the amount of video data that is produced on a daily basis is staggering. Being able to analyse these videos (e.g. to detect certain objects, actions, persons *etc*.) is important for many applications including advertising, search, content proposal and security. Furthermore, being able to do the analysis automatically is crucial, as the volume of data produced daily increases.

Computer vision is the field of science that involves processing and analysing images and videos in an automatic manner. While the majority of the computer vision research today focuses on analysing still images, video inherently offers much richer information. For instance, in video we can typically see the same object transition smoothly between different viewpoints (*e.g*. we can see a car from the front go to the left side view, and all the small variations in between). Furthermore, we can actually observe the relationships between all these slightly different viewpoints: a car cannot transition from the frontal view directly to the left side view, it has to transition trough all the intermediate viewpoints in succession. As another example, we can observe the

body motion when a human takes an action (e.g. swinging motion when a tennis player serves a ball). If we wanted to recognise the action performed by that person using a still image, we would have to rely mainly on context to take a decision: he holds a racket, he is on a tennis court. Using video, however, we can also use the motion of the player (swinging motion) in addition to the appearance cues (racket, tennis court).

This thesis focuses on methods for video processing. We will be discussing tasks at various levels: low-level processing where the output affects individual pixels of a single video (video object segmentation, chapters 2, 3), intermediate-level processing where the output involves individual frames from pairs of videos (temporal alignment, chapter 4) and higher lever processing where the output is groups of frames from multiple videos representing a visual aspect (aspect discovery, chapter 5).

The rest of this chapter is an overview of the computer vision tasks and concepts that will be touched in this thesis. We will briefly mention some related work in order to better give some context in which this thesis operates. For a more thorough description of the related work, we refer the reader to the corresponding chapters.

1.1 Video segmentation

Video segmentation is a fundamental task in computer vision that has received significant attention in recent years (*e.g.* Lee et al. (2011); Zhang et al. (2012); Stretcu and Leordeanu (2015); Giordano et al. (2015); Kong et al. (2013)). Typically, the goal of video segmentation is to separate, at the pixel-level, each frame in the video into two or more regions, called segments. Usually, each segment corresponds to a specific object in the video or the background.

Video segmentation is an important preprocessing step for many applications such as video summarisation (Lee et al., 2012), action recognition (Blank et al., 2005) and learning object class models (Prest et al., 2012), where we need to localise the object in every frame in the video. This is a very challenging task, as we may have little or no prior knowledge about the specific object's appearance, scale or position. Furthermore, the general unconstrained setting might include rapidly moving backgrounds and objects, non-rigid deformations and articulations.

Video segmentation is a broad term that encompasses a large range of different tasks. Segmentation methods vary not only in their approach but also in their level of supervision and even their desired output. In this section we will summarise and present these differences.

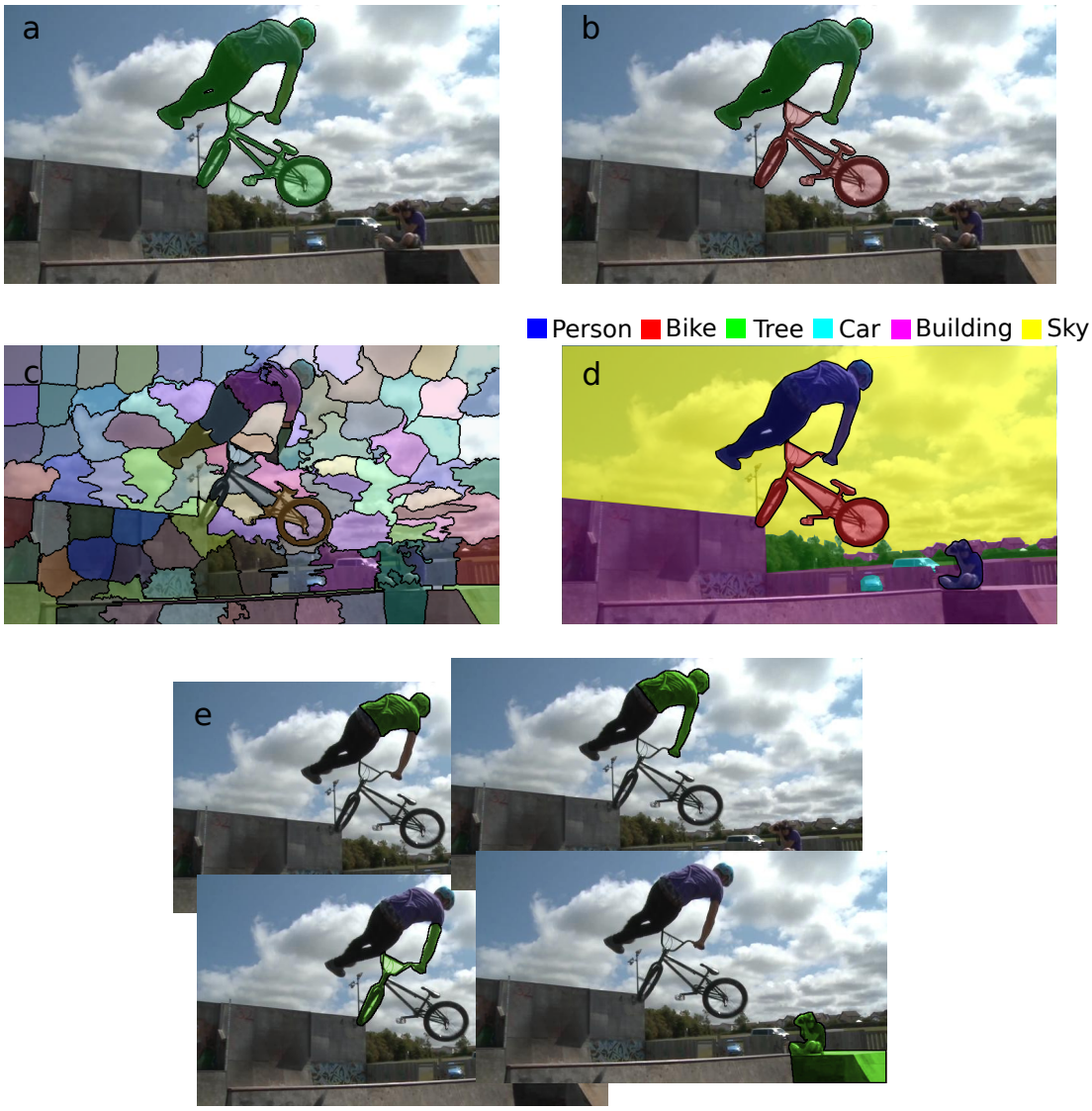

Figure 1.1: **Categories of segmentation algorithms based on expected output.** *Highlighted areas correspond to segments. Different colours within the same frame denote different segments.* ***(a)*** *Foreground object segmentation: Both foreground objects (bike, biker) are treated as a single segment.* ***(b)*** *Multi-object segmentation: each foreground object is a separate segment.* ***(c)*** *Oversegmentation: the frame is decomposed into multiple semgents. Each object might be split into multiple segments.* ***(d)*** *Semantic segmentation: the frame is decomposed into multiple segments. Each segment is given a semantic label corresponding to the depicted object class.* ***(e)*** *Video object proposals: multiple, possibly overlapping segments are proposed as candidates for the foreground objects.*

First, we discuss the differences in the desired output between different tasks. Given that video segmentation is usually meant as a preprocessing step, there are different expectations in regards to the number of segments and what they represent depending on the final application (fig. 1.1). Broadly, we can distinguish three categories of segmentation algorithms:

- **Foreground object segmentation.** In this family of works (*e.g.* Lee et al. (2011); Stretcu and Leordeanu (2015); Giordano et al. (2015); Yang et al. (2016)) the goal is to separate all foreground objects from the background. All foreground objects are treated as a single segment (foreground) and there is no attempt to separate multiple objects. This is the category that our work belongs to (chapter 2).

- **Multi-object segmentation.** In this family of works (*e.g.* Brox and Malik (2010); **?**); Ochs and Brox (2012); Kong et al. (2013); Trichet and Nevatia (2013)) the goal is to separate each object from the background and from other foreground objects. In case of multiple foreground objects, the algorithm is expected to create a separate segment per object.

- **Oversegmentation.** These works (*e.g.* Grundmann et al. (2010); Chang et al. (2013)) segment each frame to a significantly larger number of segments than the number of objects present in each frame. Each segment is relatively small and homogeneous in appearance and motion. Since the segments do not correspond to entire objects, the output is typically used as an intermediate step for other algorithms.

- **Video object proposals.** This family of works (*e.g.* Li et al. (2013); Oneata et al. (2014); Puscas et al. (2015); Wu et al. (2015)) produces multiple spatio-temporal object proposals per video, which are likely to contain the foreground object. Each video object proposal aims to contain an area that is consistent in motion and appearance through time, making it likely to contain a single object. These methods do not try to distinguish which of the object proposals correspond to foreground objects. Instead, they are intended as a pre-processing step for other foreground object and multi-object segmentation methods. This is a direct analogue to the object proposal line of works (Carreira and Sminchisescu, 2010; Endres and Hoiem, 2010) that are popular for still images.

- **Semantic segmentation.** In semantic segmentation works (*e.g.* Jain et al. (2013); Tang et al. (2013); Liu et al. (2014)) the methods are required to both segment the different objects depicted in the video and assign them a semantic label (*e.g.* car, bike, dog).

Another major difference between different segmentation methods is the level of supervision that is required by the user. We can categorise the various segmentation methods into four categories depending on the level of supervision they require (fig. 1.2):

- **Unsupervised methods.** Unsupervised methods (*e.g.* Zhang et al. (2012); Kong et al. (2013); Giordano et al. (2015)) require no interaction from the user: the method is given no knowledge about the appearance, location, scale or size of the foreground object. In order to segment the foreground objects, these methods typically rely on detecting salient motion or generic object-like properties (such as image saliency (Perazzi et al., 2012) or object proposals (Endres and Hoiem, 2010)). Our method presented in chapter 2 belongs to this category of works.

- **Co-segmentation methods (weakly supervised).** Co-segmentation methods (*e.g.* Joulin et al. (2014); Wang et al. (2014a); Rochan and Wang (2014); Kwak et al. (2015)) attempt to segment multiple videos showing the same object class simultaneously (*e.g.* a collection of car videos). These methods rely on the similarity in appearance and structure between the foreground objects in different videos in order to discover and segment them. These methods are considered weakly supervised as they require that the user provides a collection of videos of the same semantic class, but they do not require any location information about the object.

- **Interactive methods (supervised).** Interactive methods require the user to annotate some regions as foreground/background in one or more frames in the video. These annotations may be manual pixel-level segmentations (*e.g.* Jain and Grauman (2014); Nagaraja et al. (2015)), rough strokes on the foreground and background (*e.g.* Perez-Rua et al. (2015)), clicks on the foreground object (*e.g.* Palazzo et al. (2016)) or human gaze data (*e.g.* Spampinato et al. (2015)). Furthermore, the user may provide the annotation as a static input to the segmentation method (*e.g.* Jain and Grauman (2014); Perez-Rua et al. (2015)) or

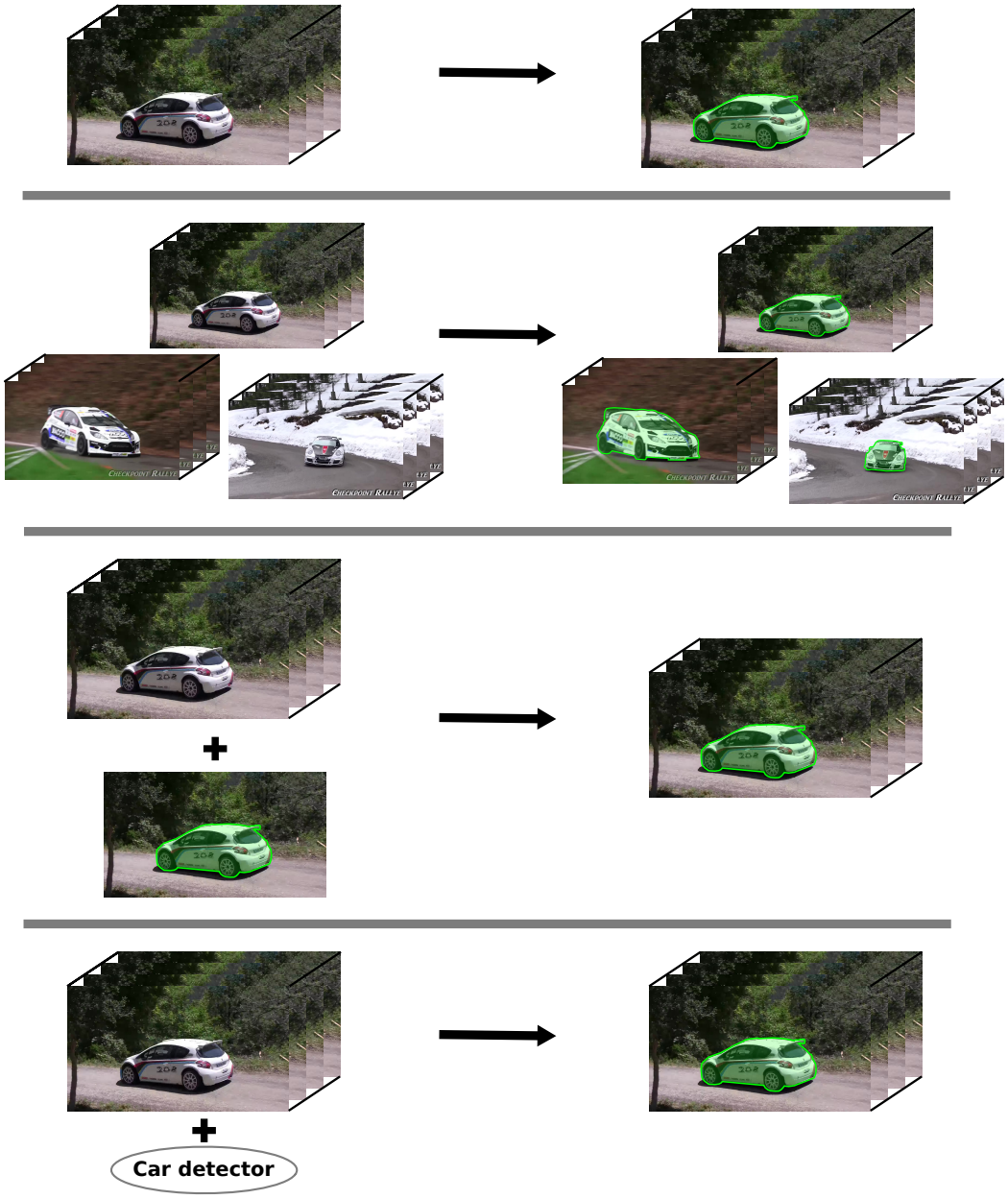

Figure 1.2: **Categories of segmentation algorithms based on the level of supervision.** *Highlighted areas correspond to foreground segments.* ***(First row)*** *Unsupervised methods: The method is given a single video, without any prior knowledge about the foreground object.* ***(Second row)*** *Co-segmentation methods: The methods are given a collection of videos showing the same object class. All the videos are then segmented jointly.* ***(Third row)*** *Interactive methods: The methods are given a single video and manual foreground/background annotations for a few frames. Here, the method is given the segmentation of the first frame as input.* ***(Fourth row)*** *Class-specific methods: The methods are given a single video and the object class of the foreground object.*

iteratively correct the segmentation using a human-in-the-loop scheme (*e.g.* Liu et al. (2008)).

- **Class-specific segmentation methods.** In class-specific methods (Zhang et al., 2015) the object class of the foreground object is known (*e.g.* the video shows a car). Although the appearance of the specific object instance shown in the video is not known a-priori, these methods utilise prior knowledge about the appearance of the object class to discover and segment the foreground object. This prior knowledge is in the form of an object detector learnt from an external training set (typically still images). Our method presented in chapter 3 belongs to this category of works.

1.2 Video temporal alignment

Video temporal alignment is the task of finding a frame-to-frame correspondence between two or more videos, so that the corresponding frames show the same image content (*e.g.* the same background or the same foreground object viewpoint, fig. 1.3). Video temporal alignment is an important computer vision task and is often a key step for several popular applications such as video morphing (Liao et al., 2014), video mosaicking and stitching (Agarwala et al., 2005), video compositing (Ruegg et al., 2013), video summarisation (Ngo et al., 2005), action recognition and video retrieval (Jiang et al., 2007) and High Dynamic Range (HDR) video (Kang et al., 2007).

The content that we want to match between the videos varies depending on the specific application. For instance, we might want to temporally align two videos of different cars so that each pair of corresponding frames show the same car viewpoint. As another example, we might want to temporally align videos of the same event (*e.g.* a music concert) taken from different cameras, so that each pair of corresponding frames shows the exact same point in time. We can group the various temporal alignment methods into three categories, based on the content they align (fig. 1.3):

- **Videos of the same dynamic scene from different viewpoints.** This category of works (*e.g.* Caspi et al. (2006); F. L. C. Padua (2009); Douze et al. (2016)) focuses on videos that show the same dynamic scene (*e.g.* concerts, sports games) taken from two or more uncalibrated cameras at different viewpoints. The goal is to temporally align the videos so that each group of corresponding frames shows the same point in time. In order to establish correspondence between the frames,

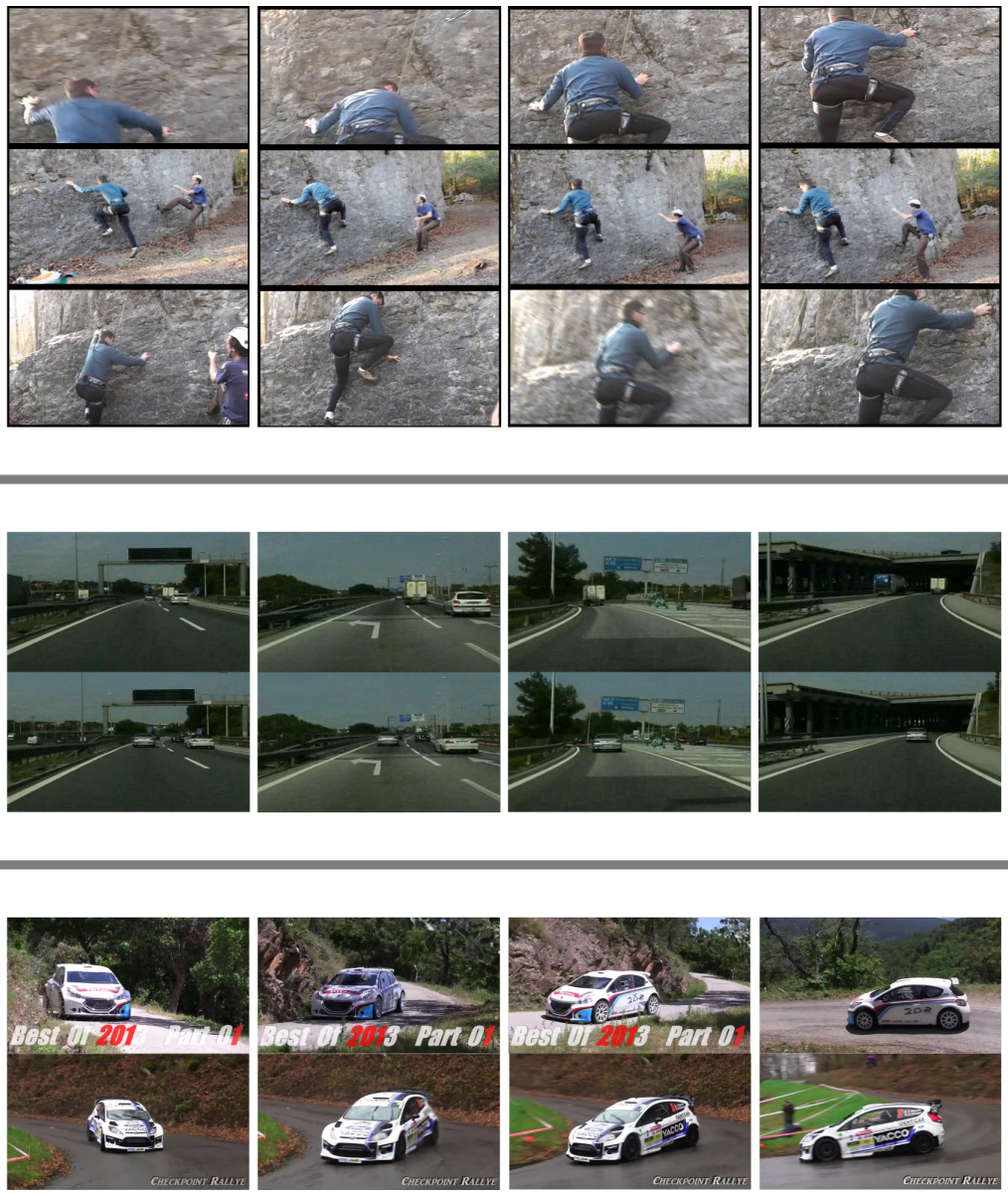

Figure 1.3: **Categories of different temporal alignment methods.** *Each row (separated by the grey lines) corresponds to a different category of alignment methods based on the content they align. Columns of frames within the same category correspond to temporally aligned frames.* ***(First row)*** *Videos of the same dynamic scene from different viewpoints: all videos show the same dynamic scene (climbing) captured by different cameras. The goal of temporal alignment is to match frames showing the same point in time.* ***(Second row)*** *Videos of the same scene at different times: the videos show the same scene (highway) taken at different points in time. The goal of temporal alignment is to match frames showing the same background.* ***(Third row)*** *Videos of semantically similar scenes: the videos show the same semantic content (two different cars racing at different race tracks). The goal of temporal alignment is to match frames showing the same object viewpoint.*

most methods typically involve a spatial alignment step in order to measure the similarity between any two frames (the frames are similar if they can be spatially aligned). As such, it is common to jointly solve both spatial and temporal alignment simultaneously.

- **Videos of the same scene at different times.** This category of works (*e.g.* Diego et al. (2013); Evangelidis and Bauckhage (2013); Wang et al. (2014b)) focuses on videos that show the same scene taken from two cameras at different times (*e.g.* a video taken from two different cars travelling down the same road). A common assumption in this line of works is that the cameras follow roughly the same trajectory. The goal is then to temporally align the videos so that each pair of corresponding frames shows the same background (*e.g.* same part of the road). Similar to the first category, these methods often jointly solve both spatial and temporal alignment jointly.

- **Videos of semantically similar scenes.** This category of works focuses on videos that show similar semantic content rather than the same scene. A typical example is videos of different people performing a specific action (*e.g.* drinking, hand waiving), where we want the corresponding frames to show the same pose. Another example would be videos of cars on race tracks, where we want the corresponding frames to show the same car viewpoint. In contrast to the first two categories, where the focus of alignment was the whole image and the background respectively, this category of works aims at aligning the foreground. This is a relatively harder task as the appearance of the objects and the backgrounds is different. Furthermore, the viewpoints may be shown a different number of times and in different order between the videos. Our method presented in chapter 4 belongs to this category of works.

These differences in the content to be aligned greatly affect the methodology to solve the task. For instance, in the case of videos of the same dynamic scene from different viewpoints, all videos show the same sequence of events. Because of that, there is exactly one correct correspondence for each video frame: the one that corresponds to the same point in time. Furthermore, the correspondences are sequential: events in one video appear in the same sequence in the other videos as well. In the case of videos of semantically similar scenes however (*e.g.* cars racing, last row of fig 1.3), there may be multiple possible correspondences for each frame (*i.e.*, the same viewpoint is shown more than once in a video). Additionally, the sequence of viewpoint transitions is not

necessarily the same in both videos and/or may not happen at the same speed, so the correspondences are not sequential. In order to accommodate these differences, there exist various approaches with different constraints suited to the task (sec. 4.2).

For videos of semantically similar scenes we are interested in aligning them with respect to the foreground objects (*e.g.* same viewpoint or pose). For that reason, it is important to focus the appearance measurement on the foreground objects rather than the background. Otherwise, the appearance measurement would be overwhelmed by information that is irrelevant to the content we want to align. For instance, whether the background is an urban area or a field has no relevance to the viewpoint of a car shown in the image. As such, measuring its appearance can only confuse an algorithm that tries to align the viewpoint of cars. In that respect, our video object segmentation technique is an important tool as it enables us to automatically localise the foreground object in each video.

1.3 Aspect discovery

Traditionally, visual aspects have been defined as distinct viewpoints of rigid 3-D objects (Koenderink and van Doorn, 1979; Plantinga and Dyer, 1986; Bowyer et al., 1988; Cyr and Kimia, 2001). A viewpoint is considered indistinguishable from another viewpoint, if the descriptors used to describe their appearance are identical. However, this definition is problematic for modern computer vision. Unlike early works, which described object appearance using edges and corners (Koenderink and van Doorn, 1979; Plantinga and Dyer, 1986; Bowyer et al., 1988; Cyr and Kimia, 2001), modern appearance descriptors such as the outputs of Convolutional Neural Networks (*e.g.* Girshick et al. (2014)) depend on many factors besides viewpoint such as illumination or intra-class appearance variation. As such, two images showing two object class instances from identical viewpoints will produce different appearance descriptors. More importantly, viewpoint alone is not enough to capture the appearance variation of complex articulated objects in natural images. For instance, an image of a tiger lying down is visually very different from one of a tiger standing up, even if both of them are showing the exact same viewpoint (fig. 1.4b, c). Hence, in this thesis we consider a broader notion of aspect that encompasses four factors of variation: viewpoint, articulated pose, occlusions and cropping by the image borders.

Aspect discovery is the task of discovering all the different aspects that are present in a collection of images showing the same object class (*e.g.* tigers fig. 1.5). While

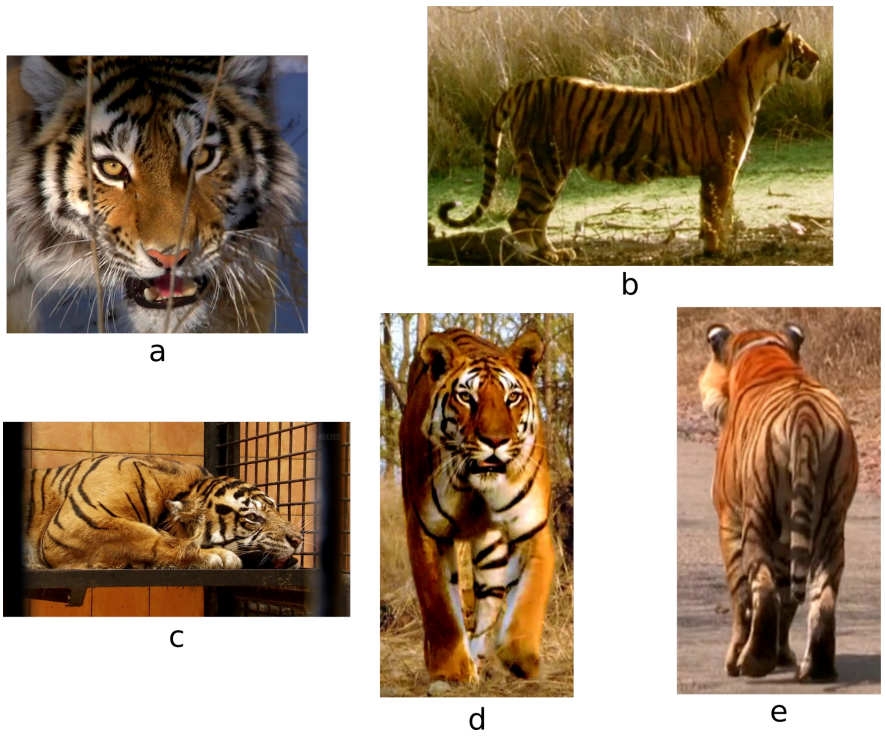

Figure 1.4: **Example aspects of the tiger class.** *In our work we consider four factors of variation: viewpoint, articulated pose, occlusions and cropping by the image borders. Notice how aspects b and c are visually very different despite showing the tigers from the same viewpoint.*

a fundamental task on its own right, aspect discovery is often an implicit first step of several works (*e.g.* Felzenszwalb et al. (2010); Dong et al. (2013); Drayer and Brox (2014)), in order to train specialised classifiers for different aspects (components of a mixture model). In order to discover the aspects, these works rely on expensive and time consuming location annotations, such as bounding-boxes around the object and keypoints. Furthermore, since aspect discovery is only a part of a larger system, the quality of the discovered aspects is never evaluated directly but through its perceived performance improvement to the overall system.

In this thesis we explore weakly-supervised aspect discovery from video which we pose as a frame clustering problem. Similar to video alignment, in order to find similar frames (which could indicate an aspect) we need to focus on the appearance of the object rather than the background. Using our unsupervised foreground object segmentation method, we are able to localise the object in the image without the need for manual annotation. Finally, we propose a protocol to evaluate the quality of our aspect discovery method directly.

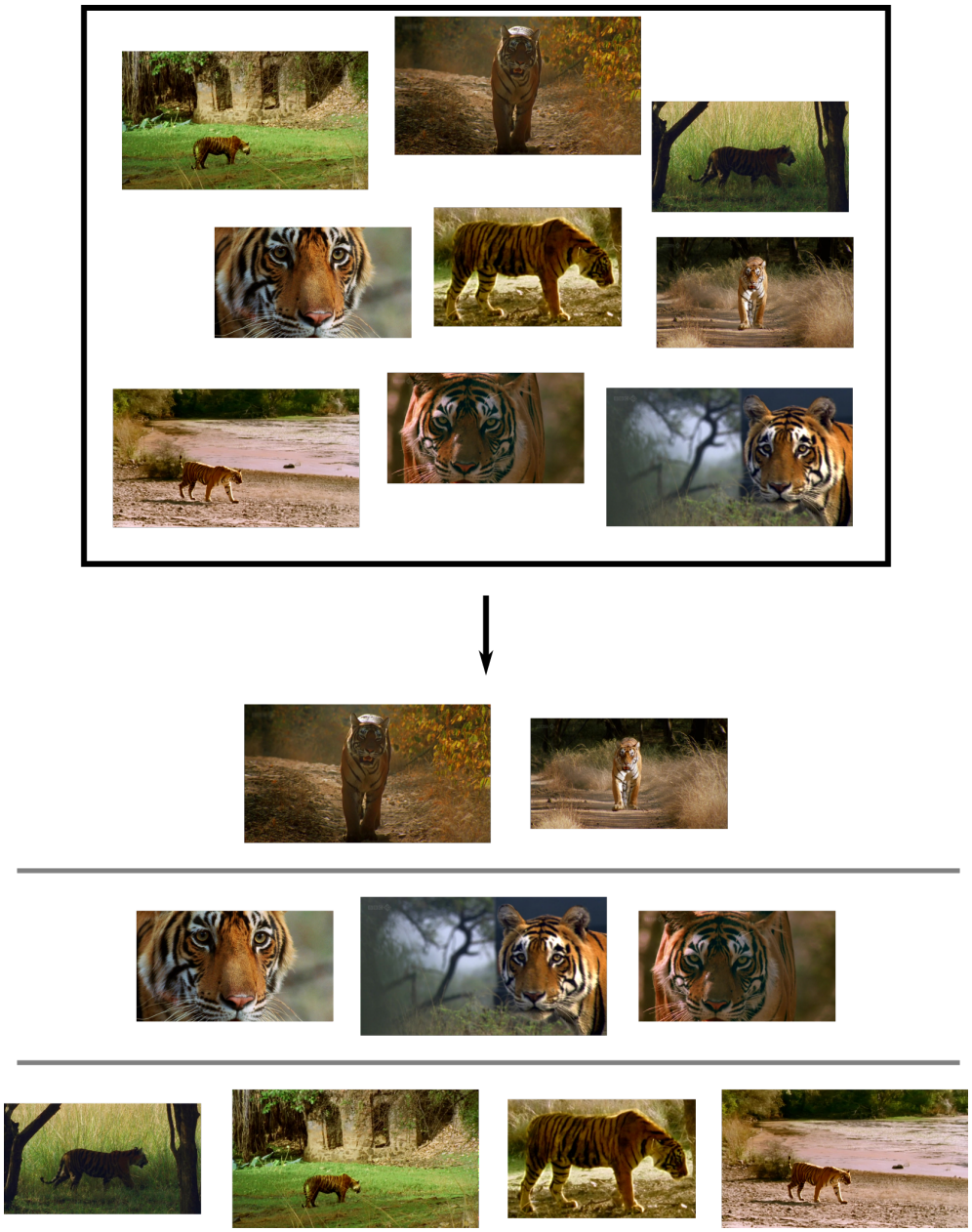

Figure 1.5: **Aspect discovery task**. *Given a collection of images of an object class (top), we want to discover all different aspects of that class that are depicted therein (bottom). Each row (bottom) indicates an aspect of the class shown in the images.*

1.4 Contributions

The main contributions of this thesis are:

- A computationally efficient unsupervised foreground object segmentation method, which is able to handle challenging unconstrained videos. The method is able to handle rapidly moving backgrounds and objects, as well as non-rigid deformations and articulations without having prior knowledge about the object's appearance, size or location. Our method relies on a novel technique to rapidly produce a rough initial segmentation based solely on motion. This initial estimate is then refined by integrating information over the entire video. This method is presented in chapter 2.

- An extension of the unsupervised foreground object segmentation method for the case of class-specific object segmentation. Our method relies on state-of-the-art object detectors in order to inject prior knowledge about the appearance of the object class. This achieves a significant improvement over the unsupervised setting. This is presented in chapter 3.

- A model and optimisation procedure for video temporal alignment of semantically similar scenes. Unlike previous works that assume that the videos show a scripted sequence of events (possibly out of phase) our method is able to cope with realistic, unconstrained videos, where events may occur in different order, occur in only one of the videos or occur multiple times in a video. We demonstrate our model on temporal alignment for object viewpoint and we propose a new evaluation protocol that is suitable for this setting. Finally, we show that our protocol corresponds to what humans perceive as a visually pleasing alignment through a substantial user study. This work is presented in chapter 4.

- An extensive exploration of weakly-supervised aspect discovery from video, which we pose as an image clustering problem. We experiment with several modern appearance descriptors and various levels of spatial support and carefully evaluate the benefits of exploiting video over still images for the task. In contrast to previous works that only evaluated aspect discovery indirectly, through its impact on other tasks, our exploration relies on a new protocol that evaluates aspect discovery directly. This work is presented in chapter 5.

Chapters 2, 4 and 5 address different problems and are self-contained, so that readers interested in a particular topic will be able to understand it without referring to other chapters. Chapter 3 extends the segmentation model presented in chapter 2 and as such, it should be read after it. Each chapter includes a small introduction to the problem to be addressed, a discussion of the related work in that particular field and the related results.

Chapter 2

Class-agnostic video object segmentation

2.1 Introduction

Video object segmentation is the task of separating, at the pixel-level, the foreground objects from the background in a video. This is a fundamental task in computer vision that has a wide range of applications such as video summarisation (Lee et al., 2012), action recognition (Blank et al., 2005) and learning object class models (Prest et al., 2012).

In this chapter we address the issue of class-agnostic video object segmentation (*i.e.*, we do not know in advance the class of the foreground objects). The task has been addressed by methods requiring a user to annotate the object position in some frames (Bai et al., 2009; Price et al., 2009; Wang and Collomosse, 2012; Tsai et al., 2010), and by fully automatic methods (Lee et al., 2011; Ochs and Brox, 2012; Wang and Wang, 2014; Stretcu and Leordeanu, 2015; Giordano et al., 2015), which input just the video. The latter scenario is more practically relevant, as a good solution would enable processing large amounts of video without human intervention. However, this task is very challenging, as the method is given no knowledge about the object appearance, scale or position. The only cue is that the foreground object moves differently to the background in at least some part of the video. Moreover, the general unconstrained setting might include rapidly moving backgrounds and objects, non-rigid deformations and articulations (fig. 2.6-2.8).

In this chapter we present a method for fully automatic video object segmentation in these unconstrained settings. Our method is computationally efficient and makes

minimal assumptions about the video: the only requirement is for the object to move differently from its surrounding background in a good fraction of the video. The object can be static in a portion of the video and only part of it can be moving in some other portion (e.g. a cat starts running and then stops to lick its paws). Our method does not require a static or slowly moving background (as opposed to classic background subtraction methods (Cucchiara et al., 2003; Barnichm and Van Droogenbroeck, 2011; Brutzer et al., 2011)). Moreover, it does not assume the object follows a particular motion model, nor that all its points move homogeneously (as opposed to methods based on clustering point tracks (Brox and Malik, 2010; **?**; Ochs and Brox, 2012)). This is especially important when segmenting non-rigid or articulated objects such as animals (fig. 2.6-2.8).

Our method works by producing a rough estimate of which pixels belong to the foreground objects. The key new element in our approach is a rapid technique to detect salient motion based on the motion boundaries in pairs of subsequent frames (sec. 2.3.1). The initial estimate is then refined by integrating information over the whole video with a spatio-temporal extension of GrabCut (Rother et al., 2004; Lee et al., 2011; Wang and Collomosse, 2012). This second stage automatically bootstraps an appearance model based on the initial foreground estimate, and uses it to refine the spatial accuracy of the segmentation and to also segment the object in frames where it does not move (sec. 2.3.2).

Part of this chapter has been published in (Papazoglou and Ferrari, 2013). The addition of saliency (sec. 2.3.3 and results in sec. 2.4) is new material.

2.2 Related work

Interactive or supervised methods. Several methods for video object segmentation require the user to manually annotate a few frames with object segmentations and then propagate these annotations to all other frames (Bai et al., 2009; Price et al., 2009; Wang and Collomosse, 2012; Jain and Grauman, 2014). Similarly, methods based on tracking (Chockalingam et al., 2009; Tsai et al., 2010; Wen et al., 2015), require the user to mark the object positions in the first frame and then track them in the rest of the video.

Other methods require weaker supervision from the user. The methods of (Perez-Rua et al., 2015; Nagaraja et al., 2015) require a user to annotate with a few strokes the foreground and background regions in the first video frame. These strokes are

propagated in the video and refined using motion and appearance cues. In (Palazzo et al., 2016) the authors present annotation as an online game, where the annotators need to click on the foreground object as the video plays. The object is then segmented using a typical energy optimisation approach. In (Spampinato et al., 2015) the authors follow a similar approach but use human gaze data as cues instead of user clicks.

Background subtraction. Classic background subtraction methods model the appearance of the background at each pixel and consider pixels that change rapidly to be foreground. These methods typically assume a stationary, or slowly panning camera (Cucchiara et al., 2003; Barnichm and Van Droogenbroeck, 2011; Brutzer et al., 2011). The background should change slowly in order for the model to update safely without generating false-positive foreground detections. These assumptions are generally true for applications such as video surveillance, but do not apply to unconstrained video where there is fast camera motion.

Clustering point tracks. Several automatic video segmentation methods track points over several frames and then cluster the resulting tracks based on pairwise (Brox and Malik, 2010; **?**; Kong et al., 2013; Trichet and Nevatia, 2013) or triplet (Ochs and Brox, 2012) similarity measures. The underlying assumption induced by pairwise clustering (Brox and Malik, 2010; **?**; Kong et al., 2013; Trichet and Nevatia, 2013) is that all object points move according to a single translation, while the triplet model (Ochs and Brox, 2012) assumes a single similarity transformation. These assumptions have trouble accommodating non-rigid or articulated objects. Our method instead does not attempt to cluster object points and does not assume any kind of motion homogeneity. The object only needs to move sufficiently differently from the background to generate motion boundaries along most of its physical boundary. On the other hand, these methods (Brox and Malik, 2010; **?**; Ochs and Brox, 2012; Kong et al., 2013; Trichet and Nevatia, 2013) try to place multiple objects in separate segments, whereas our method produces a simpler binary segmentation (all objects vs background).

Ranking object proposals. The works (Lee et al., 2011; Ma and Latecki, 2012; Zhang and Shah, 2013; Wang and Wang, 2014, 2016; Stretcu and Leordeanu, 2015) are closely related to ours, as they tackle the very same task. These methods are based on finding recurring object-like segments, aided by recent techniques for measuring generic object appearance (such as (Endres and Hoiem, 2010)), and achieve impres-

sive results on the SegTrack benchmark (Tsai et al., 2010). While the object proposal infrastructure is necessary to find out which image regions are objects vs background, it makes these methods very slow (minutes/frame). In our work instead, this goal is achieved by a much simpler, faster process (sec. 2.3.1).

Refining rough foreground segmentation. Following our original work in (Papazoglou and Ferrari, 2013) several methods (Giordano et al., 2015; Wang et al., 2015; Yang et al., 2016) have adopted our scheme: they use motion to find a rough initial segmentation, learn an appearance model from it and finally segment the video using an energy formulation that promotes spatio-temporal smoothness.

The works of (Wang et al., 2015; Giordano et al., 2015) innovate on how the derive the rough initial segmentation. In (Wang et al., 2015) the authors propose a saliency measure designed for video to produce the initial rough segmentation. This saliency measure is produced using geodesic distance from spatial edges and motion boundaries. This leads to more efficient inference during the optimisation procedure. Finally, (Giordano et al., 2015) propose a method to produce the initial rough segmentation without the need for computing optical flow. Instead, they compare the output of a superpixel method in subsequent frames to detect motion. This leads to reduced computation times, as optical flow is a major bottleneck for many algorithms. In (Yang et al., 2016) the authors innovate on the refinement stage by embedding the appearance model learning and segmentation refinement in a single energy formulation and simultaneously optimise both, which leads to faster inference times.

The work of (Faktor and Irani, 2014) proposes non-local concensus voting of reoccuring regions across the video to iteratively refine a saliency map at every frame. In contrast to other methods in this category, they do not refine this saliency map using an energy formulation, but instead use a single saliency threshold to derive the foreground regions.

Co-segmentation. Various methods attempt to simultaneously segment all videos belonging to the same semantic class (Joulin et al., 2014; Wang et al., 2014a; Rochan and Wang, 2014; Kwak et al., 2015). The common approach of these methods is to create a pool of candidate object proposals for every frame of every video. The assumption is that each frame contains exactly one instance of the foreground object, so the segmentation problem becomes finding the object proposal in each frame that corresponds to the foreground object. These methods then exploit the appearance and

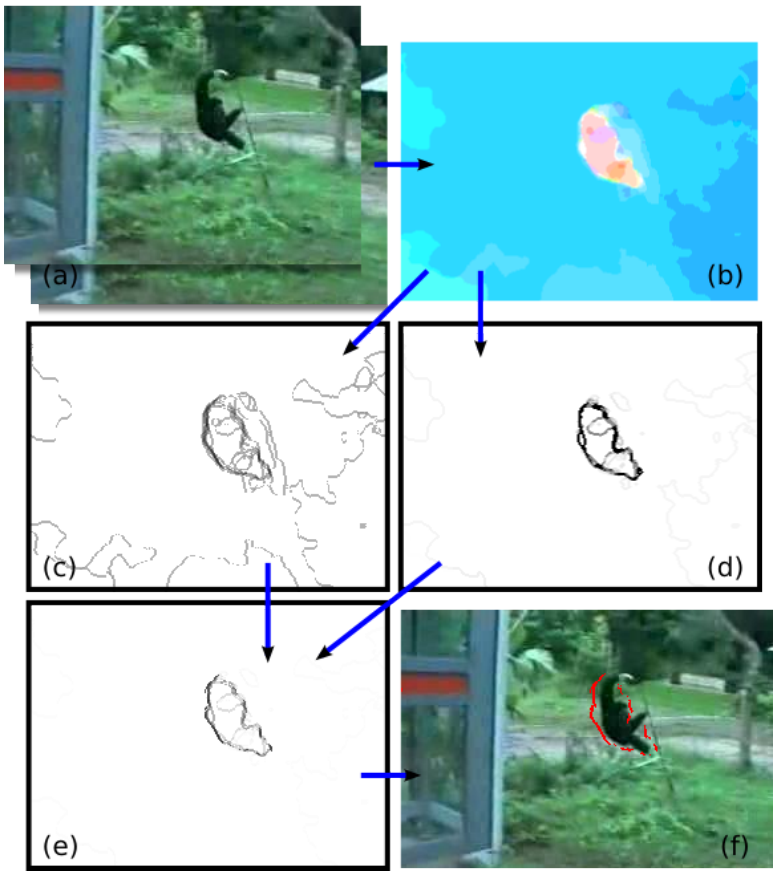

Figure 2.1: **Motion boundaries.** *(a) Two input frames. (b) Optical flow $\vec{f}_p$. The hue of a pixel indicates its direction and the color saturation its velocity. (c) Motion boundaries b_p^m, based on the magnitude of the gradient of the optical flow. (d) Motion boundaries b_p^θ, based on difference in direction between a pixel and its neighbours. (e) Combined motion boundaries b_p. (f) Final, binary motion boundaries after thresholding, overlaid on the first frame.*

structure similarity between the foreground objects in the different videos in order to discover and segment them.

The method of (Wang et al., 2014a) does not assume that all frames contain the foreground object. However it requires that a user marks a few frames that contain the foreground object.

2.3 Overview of the method

The goal of our work is to segment objects that move differently than their surroundings. Our method has two main stages: (1) efficient initial foreground estimation

(sec. 2.3.1), (2) foreground-background labelling refinement (sec. 2.3.2). We now give a brief overview of these two stages, and then present them in more detail in the rest of the section.

(1) Efficient initial foreground estimation. The goal of the first stage is to rapidly produce an initial estimate of which pixels might be inside the object *based purely on motion* (inside-outside maps). We compute the optical flow between pairs of subsequent frames and detect motion boundaries. Ideally, the motion boundaries will form a complete closed curve coinciding with the object boundaries. However, due to inaccuracies in the flow estimation, the motion boundaries are typically incomplete and do not align perfectly with object boundaries (fig. 2.1f). Also, occasionally false positive boundaries might be detected. We propose a novel, computationally efficient algorithm to robustly determine which pixels reside inside the moving object, taking into account all these sources of error (fig. 2.2c).

(2) Foreground-background labelling refinement. As they are purely based on motion boundaries, the inside-outside maps produced by the first stage typically only approximately indicate where the object is. They do not accurately delineate object outlines. Furthermore, (parts of) the object might be static in some frames, or the inside-outside maps may miss it due to incorrect optical flow estimation.

The goal of the second stage is to refine the spatial accuracy of the inside-outside maps and to segment the whole object in all frames. To achieve this, it integrates the information from the inside-outside maps over all frames by (1) encouraging the spatio-temporal smoothness of the output segmentation over the whole video; (2) building dynamic *appearance models* of the object and background under the assumption that they change smoothly over time. Incorporating appearance cues is key to achieving a finer level of detail, compared to using only motion. Moreover, after learning the object appearance in the frames where the inside-outside maps found it, the second stage uses it to segment the object in frames where it was initially missed (e.g. because it is static).

2.3.1 Efficient initial foreground estimation

Optical flow. We begin by computing optical flow between pairs of subsequent frames $(t, t+1)$ using the excellent optical flow estimation algorithm of (Brox and Malik, 2010; Sundaram et al., 2010). It supports large displacements between frames and

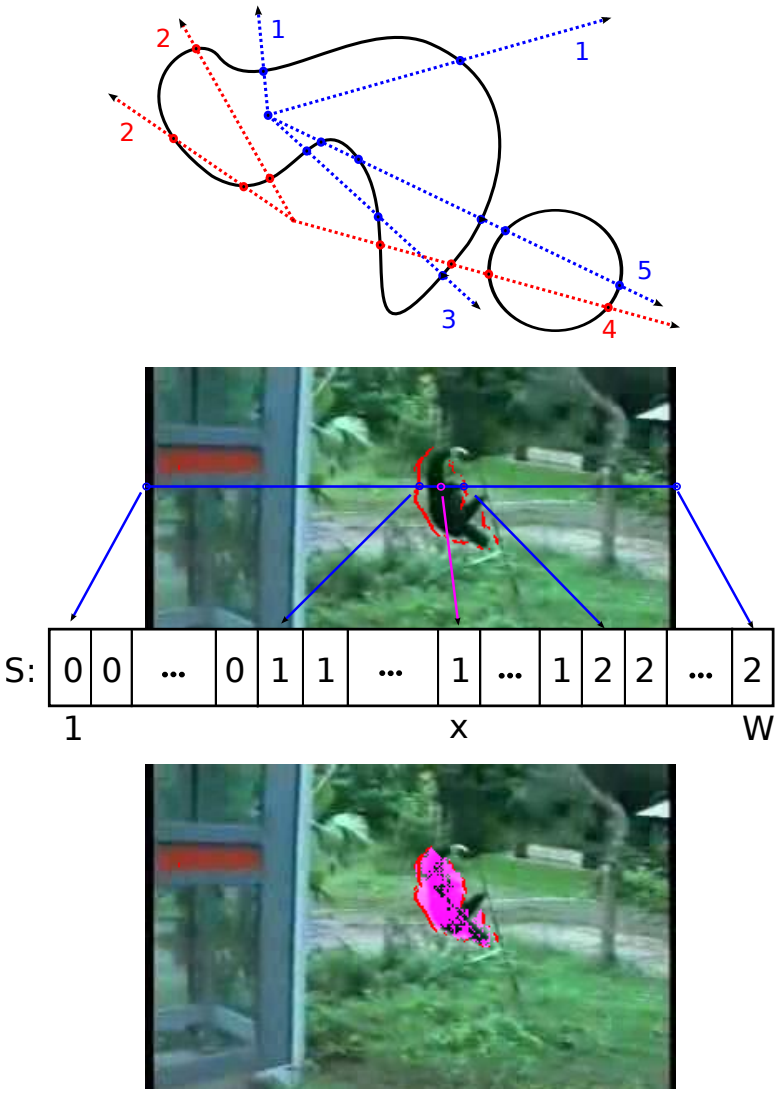

Figure 2.2: **Inside-outside maps.** *(Top) The ray-casting observation. Any ray originating inside a closed curve intersects it an odd number of time. Any ray originating outside intersects it an even number of times. This holds for any number of closed curves in the image. (Middle) Illustration of the integral intersections data structure S for the horizontal direction. The number of intersections for the ray going from pixel x to the left border can be easily computed as* $X_{\text{left}}(x,y) = S(x-1,y) = 1$*, and for the right ray as* $X_{\text{right}}(x,y) = S(W,y) - S(x,y) = 1$*. In this case, both rays vote for x being inside the object. (Bottom) The output inside-outside map* M^t*.*

has a computationally very efficient GPU implementation (Sundaram et al., 2010) (fig. 2.1a+b).

Motion boundaries. We base our approach on motion boundaries, i.e. image points where the optical flow field changes abruptly. Motion boundaries reveal the location of occlusion boundaries, which very often correspond to physical object boundaries (Sundberg et al., 2011).

Let $\vec{f}_p$ be the optical flow vector at pixel p. The simplest way to estimate motion boundaries is by computing the magnitude of the gradient of the optical flow field:

$$b_p^m = 1 - \exp(-\lambda^m ||\nabla \vec{f}_p||) \tag{2.1}$$

where $b_p^m \in [0,1]$ is the strength of the motion boundary at pixel p; λ^m is a parameter controlling the steepness of the function.

While this measure correctly detects boundaries at rapidly moving pixels, where b_p^m is close to 1, it is unreliable for pixels with intermediate b_p^m values around 0.5, which could be explained either as boundaries or errors due to inaccuracies in the optical flow (fig. 2.1c). To disambiguate between those two cases, we compute a second estimator $b_p^\theta \in [0,1]$, based on the difference in direction between the motion of pixel p and its neighbours $\mathcal{N}$:

$$b_p^\theta = 1 - \exp(-\lambda^\theta \max_{q \in \mathcal{N}} (\delta\theta_{p,q}^2)) \tag{2.2}$$

where $\delta\theta_{p,q}$ denotes the angle between $\vec{f}_p$ and $\vec{f}_q$. The idea is that if n is moving in a different direction than all its neighbours, it is likely to be a motion boundary. This estimator can correctly detect boundaries even when the object is moving at a modest velocity, as long as it goes in a different direction than the background. However, it tends to produce false-positives in static image regions, as the direction of the optical flow is noisy at points with little or no motion (fig. 2.1d).

As the two measures above have complementary failure modes, we combine them into a measure that is more reliable than either alone (fig. 2.1e):

$$b_p = \begin{cases} b_p^m, & \text{if } b_p^m > T \\ b_p^m \cdot b_p^\theta, & \text{if } b_p^m \leq T, \end{cases} \tag{2.3}$$

where T is a high threshold, above which b_p^m is considered reliable on its own. As a last step we threshold b_p at 0.5 to produce a binary motion boundary labelling (fig. 2.1f).

Inside-outside maps. The produced motion boundaries typically do not completely cover the whole object boundary. Moreover, there might be false positive boundaries, due to inaccurracy of the optical flow estimation. We present here a computationally efficient algorithm to robustly estimate which pixels are inside the object while taking into account these sources of error.

The algorithm estimates whether a pixel is inside the object based on the point-in-polygon problem (Foley et al., 1990) from computational geometry. The key observation is that any ray starting from a point inside the polygon (or any closed curve) will intersect the boundary of the polygon an odd number of times. Instead, a ray starting from a point outside the polygon will intersect it an even number of times (figure 2.2a). Since the motion boundaries are typically incomplete, a single ray is not sufficient to determine whether a pixel lies inside the object. Instead, we get a robust estimate by shooting 8 rays spaced by 45 degrees. Each ray casts a vote on whether the pixel is inside or outside. The final inside-outside decision is taken by majority rule, i.e. a pixel with 5 or more rays intersecting the boundaries an odd number of times is deemed inside.

Realizing the above idea with a naive algorithm would be computationally expensive (i.e. quadratic in the number of pixels in the image). We propose an efficient algorithm which we call *integral intersections*, inspired by the use of integral images in (Viola and Jones, 2001). The key idea is to create a special data structure that enables very fast inside-outside evaluation by massively reusing the computational effort that went into creating the data structure.

For each direction (horizontal, vertical and the two diagonals) we create a matrix S of the same size $W \times H$ as the image. An entry $S(x,y)$ of this matrix indicates the number of boundary intersections along the line going from the image border up to pixel (x,y). For simplicity, we explain here how to build S for the horizontal direction. The algorithm for the other directions is analogous. The algorithm builds S one line y at a time. The first pixel $(1,y)$, at the left image border, has value $S(1,y) = 0$. We then move rightwards one pixel at a time and increment $S(x,y)$ by 1 each time we transition from a non-boundary pixel to a boundary pixel. This results in a line $S(:,y)$ whose entries count the number of boundary intersections (fig. 2.2b.).

After computing S for all horizontal lines, the data structure is ready. We can now determine the number of intersections X for both horizontal rays (left→right,

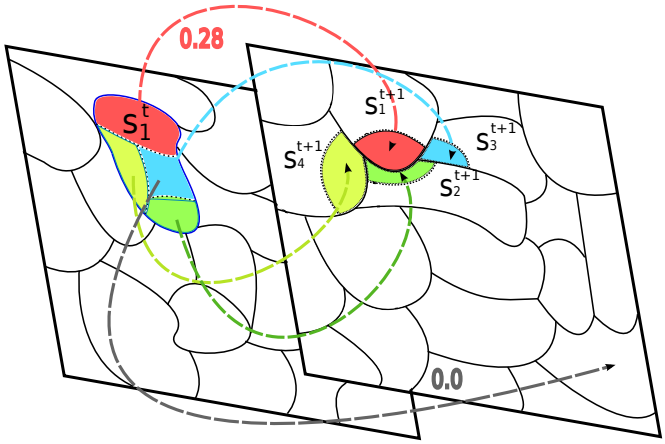

Figure 2.3: **Example connectivity $\mathcal{E}_t$ over time.** *Superpixel s_1^t contains pixels that lead to s_1^{t+1}, s_2^{t+1}, s_3^{t+1}, s_4^{t+1}. As an example, the weight $\phi(s_1^t, s_1^{t+1})$ is* 0.28 *(all others are omitted for clarity).*

right→left) emanating from a pixel (x,y) in *constant time* by

$$X_{\text{left}}(x,y) = S(x-1,y) \tag{2.4}$$

$$X_{\text{right}}(x,y) = S(W,y) - S(x,y) \tag{2.5}$$

where W is the width of the image, i.e. the rightmost pixel in a line (fig. 2.2b).

Our algorithm visits each pixel exactly once per direction while building S, and once to compute its vote, and is therefore *linear* in the number of pixels in the image. The algorithm is very fast in practice and takes about 0.1s per frame of a HD video (1280x720 pixels) on a modest CPU (Intel Core i7 at 2.0GHz).

For each video frame t, we apply the algorihtm on all 8 directions and use majority voting to decide which pixels are inside, resulting in an inside-outside map M^t (fig. 2.2c).

2.3.2 Foreground-background labelling refinement

We formulate video segmentation as a pixel labelling problem with two labels (foreground and background). We oversegment each frame into superpixels $\mathcal{S}^t$ (Achanta et al., 2012), which greatly reduces computational efficiency and memory usage, enabling to segment much longer videos.

Each superpixel $s_i^t \in \mathcal{S}^t$ can take a label $l_i^t \in \{0,1\}$. A labelling $\mathcal{L} = \{l_i^t\}_{t,i}$ of all superpixels in all frames represents a segmentation of the video. Similarly to other seg-

mentation works (Lee et al., 2011; Rother et al., 2004; Wang and Collomosse, 2012), we define an energy function to evaluate a labeling:

$$E(\mathcal{L}) = \sum_{t,i} U_i^t(l_i^t) + \alpha_V \sum_{(i,j,t)\in\mathcal{E}_s} V_{ij}^t(l_i^t, l_j^t) + \alpha_W \sum_{(i,j,t)\in\mathcal{E}_t} W_{ij}^t(l_i^t, l_j^{t+1}) \tag{2.6}$$

The pairwise potentials V and W encourage spatial and temporal smoothness, respectively, while the scalars α weight the various terms. The unary term U^t symbolises the sum of all individual unary terms that we use. In this chapter we will be experimenting with two possible unary potentials based on appearance and location.

$$U_i^t(l_i^t) = A_i^t(l_i^t) + \alpha_L L_i^t(l_i^t) \tag{2.7}$$

where A^t is a unary potential evaluating how likely a superpixel is to be foreground or background according to the appearance model of frame t. The second unary potential L^t is based on a location prior model encouraging foreground labellings in areas where independent motion has been observed. As we explain in detail later, we derive both the appearance model and the location prior parameters from the inside-outside maps M^t.

The output segmentation is the labeling that minimises (2.6):

$$\mathcal{L}^* = \underset{\mathcal{L}}{\operatorname{argmin}} E(\mathcal{L}) \tag{2.8}$$

As E is a binary pairwise energy function with submodular pairwise potentials, we minimise it exactly with graph-cuts. Next we use the resulting segmentation to re-estimate the appearance models and iterate between these two steps, as in GrabCut (Rother et al., 2004). Below we describe the potentials in detail.

Smoothness V, W. The spatial smoothness potential V is defined over the edge set $\mathcal{E}_s$, containing pairs of spatially connected superpixels. Two superpixels are spatially connected if they are in the same frame and are adjacent.

The temporal smoothness potential W is defined over the edge set $\mathcal{E}_t$, containing pairs of temporally connected superpixels. Two superpixels s_i^t, s_j^{t+1} in subsequent frames are connected if there at least one pixel of s_i^t that moves into s_j^{t+1} according to the optical flow (fig. 2.3).

The functions V, W are standard contrast-modulated Potts potentials (Rother et al., 2004; Wang and Collomosse, 2012; Lee et al., 2011):

$$V_{ij}^t(l_i^t, l_j^t) = \mathrm{dis}(s_i^t, s_j^t)^{-1} [l_i^t \neq l_j^t] \exp(-\beta \mathrm{col}(s_i^t, s_j^t)^2) \tag{2.9}$$

$$W_{ij}^t(l_i^t, l_j^{t+1}) = \phi(s_i^t, s_j^{t+1})[l_i^t \neq l_j^t] \exp(-\beta \text{col}(s_i^t, s_j^{t+1})^2) \tag{2.10}$$

where dis is the Euclidean distance between the centres of two superpixels and col is the difference between their average RGB color. The factor that differs from the standard definition is ϕ, which is the percentage of pixels within the two superpixels that are connected by the optical flow. This is a better weight than the Euclidean distance, as it is invariant of the speed of the motion.

Appearance model $\mathbf{A^t}$. The appearance model consists of two Gaussian Mixture Models over RGB colour values[1], one for the foreground (fg) and one for the background (bg). In the task of interactive segmentation (Rother et al., 2004), where this methodology originated, the appearance model parameters are estimated from some manually labelled pixels. In this paper instead, we estimated them automatically based on the inside-outside maps M^t (sec. 2.3.1).

We estimate appearance models A^t for each frame t. However, since the appearance of the fg and bg typically changes smoothly over time, these models are tightly coupled as their estimation integrates information over the whole video. Hence, the collection of per-frame models can be seen as a single *dynamic appearance model*.

At each frame t we estimate a fg model from all superpixels in the video, weighted by how likely they are to be foreground and by how close in time they are to t. More precisely, the weight of each superpixel $s_i^{t'}$ in frame t' is

$$\exp(-\lambda^A \cdot (t - t')^2) \cdot r_i^{t'} \tag{2.11}$$

The first factor discounts the weight of $s_i^{t'}$ over time. The second factor is the percentage of pixels of $s_i^{t'}$ that are inside the object according to the inside-outside map $M^{t'}$. The estimation of bg appearance models is analogous, with the second factor replaced by $1 - r_i^{t'}$ (i.e. the ratio of pixels considered to be outside the object).

After estimating the foreground-background appearance models, the unary potential $A_i^t(l_i^t)$ is the log-probability of s_i^t to take label l_i^t under the appropriate model (i.e. the foreground model if $l_i^t = 1$ and the background one otherwise).

Having these appearance models in the segmentation energy (2.6) enables to segment the object more accurately than possible from motion alone, as motion estimation is inherently inaccurate near occlusion boundaries. Moreover, the appearance models are integrated over large image regions and over many frames, and therefore can robustly estimate the appearance of the object, despite faults in the inside-outside maps.

[1] As the basic units are superpixels, all measurements refer to their average RGB value.

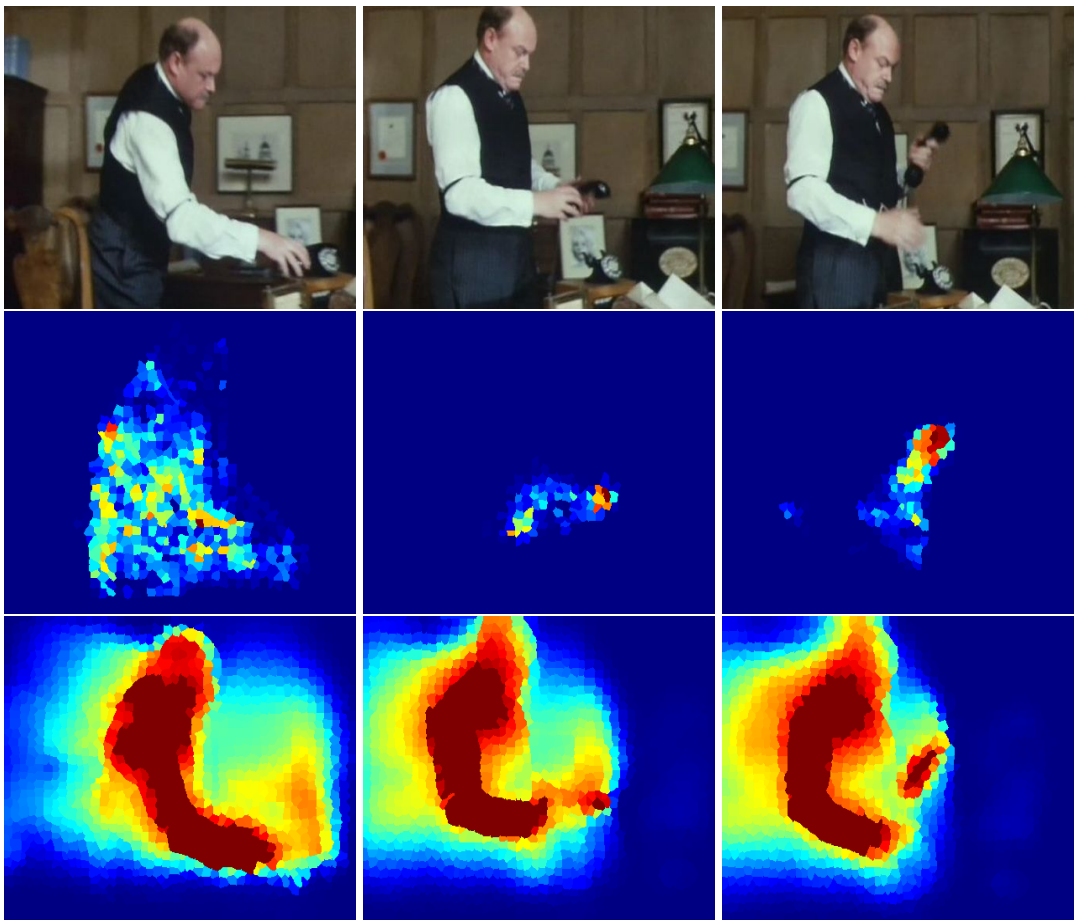

Figure 2.4: **Location model.** *Top row: three video frames. Middle row: likelihood of foreground based on the inside-outside maps in individual frames. They miss large parts of the person in the second and third frames, as the head and torso are not moving. Bottom row: the location model based on propagating the inside-outside maps. It includes most of the person in all frames.*

The appearance models then transfer this knowledge to other positions within a frame and to other frames, by altering towards foreground the unary potential of pixels with object-like appearance, even if the inside-outside maps missed them. This enables completing the segmentation in frames where only part of the object is moving, and helps segmenting it even in frames where it does move at all.

Location model $\mathbf{L}^t$. When based only on appearance, the segmentation could be distracted by background regions with similar colour to the foreground (even with perfect appearance models). Fortunately, the inside-outside maps can provide a valuable *location prior* to anchor the segmentation to image areas likely to contain the object,

as they move differently from the surrounding region. However, in some frames (part of) the object may be static, and in others the inside-outside map might miss it because of incorrect optical flow estimation (fig. 2.4, middle row). Therefore, directly plugging the inside-outside maps as unary potentials in L^t would further encourage an all-background segmentation in frames where they missed the object.

We propose here to *propagate* the per-frame inside-outside maps over time to build a more complete location prior L^t. The key observation is that 'inside' classifications are more reliable than 'outside' ones: the true object boundaries might not form a near-closed motion boundary due to the reasons above, but accidental near-closed boundaries rarely form out of noise. Therefore, our algorithm *accumulates* inside points over the entire video sequence, following the optical flow (fig. 2.4, bottom row).

The algorithm proceeds recursively. The value of the location prior at a superpixel s_i^t is initially $L_i^t := r_i^t$, i.e. the percentage of its pixels that are inside the object according to the inside-outside map M^t. We start propagating from frame 1 to frame 2, then move to frame 3 and so on. At each step, the value of the location prior for a superpixel s_j^{t+1} in frame $t+1$ gets updated to

$$L_j^{t+1} := L_j^{t+1} + \gamma \frac{\sum_i \phi(s_i^t, s_j^{t+1}) \cdot \psi(s_i^t) \cdot L_i^t}{\sum_i \phi(s_i^t, s_j^{t+1})} \tag{2.12}$$

where the summation runs over all superpixels in frame t; the connection weight ϕ is the percentage of pixels in superpixel s_i^t that connect to superpixel s_j^{t+1} by following the optical flow (fig. 2.3); $\gamma \in [0,1]$ controls the rate of accumulation; ψ is a transfer quality measure, down-weighting propagation if the optical flow for s_i^t is deemed unreliable

$$\psi(s_i^t) = \exp(-\lambda^\psi \sum_{p \in s_i^t} ||\nabla \vec{f}_p||) \tag{2.13}$$

In essence, ψ measures the sum of the flow gradients in s_i^t; large gradients can indicate depth discontinuities, where the optical flow is often inaccurate, or that s_i^t might cover bits of two different objects.

We run the forward propagation step above and an analogous backward step, starting from the last frame towards the first one. These two steps are run independently. The final location prior L^t is the normalised sum of the two steps.

2.3.3 Adding image saliency

Our model so far relied solely on the inside-outside maps to drive the refinement process: the location prior model is computed directly from them and the appearance

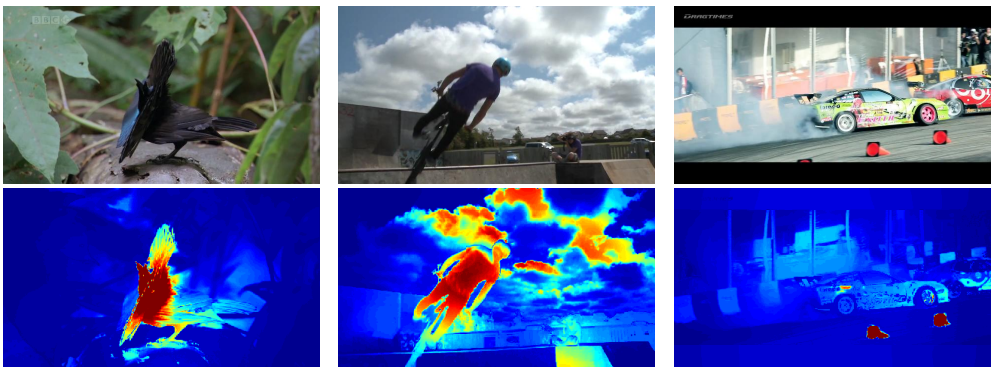

Figure 2.5: **Examples of image saliency.** *Top row: three video frames. Bottom row: a heatmap visualisation of the image saliency measure of (Perazzi et al., 2012). Red corresponds to high saliency values while blue corresponds to low saliency values. Image saliency can be a good indicator of the location of foreground object. Note, however, that it can sometimes fail in cluttered scenes showing multiple objects.*

model is initialised on them. These inside-outside maps and the resulting location prior intend to capture regions whose motion is discontinuous to the background, and as such they constitute a motion saliency (Sharif et al., 2008) estimate.

Here we add image saliency (Perazzi et al., 2012) as an additional, independent signal to further refine the final output. Image saliency estimators try to identify how likely each pixel is to catch the attention of a human observer. Intuitively, foreground objects are more salient than the background so an image saliency estimator can help steer the segmentation away from background regions.

The image saliency estimator of (Perazzi et al., 2012) relies on contrast to produce a saliency score using the following procedure. First, it segments the image into small, homogeneous regions (superpixels). Then it applies two Gaussian filtering steps to compute the uniqueness and spatial compactness of these regions in the image. These two measures are combined through a third Gaussian filtering step to produce the final pixel-level saliency score.

To incorporate it into our energy formulation (eq. 2.6) we add an additional unary term N^t as:

$$U_i^t(l_i^t) = A_i^t(l_i^t) + \alpha_L L_i^t(l_i^t) + \alpha_N N_i^t(l_i^t) \tag{2.14}$$

where N_i^t is the average image saliency of the pixels in superpixel s_i^t.

2.4 Experimental evaluation

We evaluate our method on three datasets: SegTrack v1 (Tsai et al., 2010), SegTrack v2 (Li et al., 2013) and YouTube-Objects (Prest et al., 2012). The parameters $\lambda, T, \beta, \gamma$ are set as detailed below.

2.4.1 SegTrack v1

Dataset. The original SegTrack dataset (Tsai et al., 2010) was first introduced to evaluate tracking algorithms, and it was adopted to benchmark video object segmentation by (Lee et al., 2011). It contains 6 videos (monkeydog, girl, birdfall, parachute, cheetah, penguin) and pixel-level ground-truth for the prominent foreground object in every frame. The dataset offers various challenges, including objects of similar color to the background, non-rigid deformations, and fast camera motion (fig. 2.6).

Setup. As in (Lee et al., 2011; Tsai et al., 2010), we quantify performance with the number of wrongly labeled pixels, averaged over all frames of a video. We set the weights α of the energy function (2.6) using two-fold cross-validation. We split the dataset into two sets of 3 and 2 videos respectively, and train the α weights in each set. When testing our method on the videos in one set, we use the weights trained on the other.

We compare to several methods (Brox and Malik, 2010; Ochs and Brox, 2012; Barnichm and Van Droogenbroeck, 2011; Lee et al., 2011; Ma and Latecki, 2012; Zhang and Shah, 2013). The video object segmentation method of (Lee et al., 2011) returns a ranked list of spatio-temporal segments likely to be objects. For the purposes of this evaluation we report the performance of the top ranked segment. Notice that this differs from what was published in (Lee et al., 2011), where the segments in which the final segmentation was manually selected out of the four top ranked segments. In contrast, our method directly returns a single foreground segment, as it discovers the foreground object automatically. We also report the results of another two methods based on ranking object proposals (Ma and Latecki, 2012; Zhang and Shah, 2013).

We also compare to more recent works (Jain and Grauman, 2014; Wang et al., 2014a; Wang and Wang, 2014; Giordano et al., 2015) that appeared after the publication of our method (Papazoglou and Ferrari, 2013). The segmentation methods of (Jain and Grauman, 2014; Wang et al., 2014a) are interactive segmentation methods. The method of (Jain and Grauman, 2014) requires a human annotator to provide a

pixel-level segmentation of the first frame in the video. The work of (Wang et al., 2014a) requires a few frames containing the foreground object to be marked by a human annotator. The method of (Wang and Wang, 2014) is based on ranking object proposals.

The recent works of (Wang et al., 2015; Giordano et al., 2015) follow our segmentation scheme (sec. 2.2). The method of Giordano et al. (2015), creates an initial rough segmentation that is later refined using an MRF energy formulation. This rough segmentation however is not derived from optical flow but by comparing the superpixel oversegmentations Achanta et al. (2012) of subsequent frames. The method of (Wang et al., 2015) creates an initial segmentation from optical edges and motion boundaries using geodesic distance.

Finally, we compare to a state-of-the-art background subtraction method (Barnichm and Van Droogenbroeck, 2011) and with two modern clustering point tracks based methods (Brox and Malik, 2010; Ochs and Brox, 2012). We used the implementations provided by the respective authors[2]. As the latter are designed to return multiple segments, we report results for the segment best matching the ground-truth segmentation.

Results. As table 2.1 shows, the addition of image saliency does not increase the performance for this dataset. It seems that the image saliency estimator we use (Perazzi et al., 2012), does not consistently give high scores to the foreground pixels for these videos. We believe that this may be due to the videos being blurry and containing a large number of artifacts. This may confuse the image saliency algorithm, as it relies heavily on contrast to evaluate the uniqueness of a region.

As table 2.1 shows, even the recent background-subtraction method (Barnichm and Van Droogenbroeck, 2011) performs poorly on this data, since it cannot handle fast camera motion. The point clustering methods (Brox and Malik, 2010; Ochs and Brox, 2012) produce better results, as they can better cope with these conditions.

Our method considerably outperforms (Brox and Malik, 2010; Barnichm and Van Droogenbroeck, 2011; Ochs and Brox, 2012) in all videos, as it handles non-rigid objects better, and tightly integrates appearance along with motion as segmentation cues. Overall, our performance is similar with (Lee et al., 2011) on some videos (*birdfall, parachute*), slightly worse on *girl* and significantly better on others (*cheetah, monkey*). This is

[2] `http://www2.ulg.ac.be/telecom/research/vibe/`
`http://lmb.informatik.uni-freiburg.de/resources/software.php`

precision	birdfall	cheetah	girl	monkey	parachute
(Barnichm and Van Droogenbroeck, 2011)	606	11210	26409	12662	40251
(Brox and Malik, 2010)	468	1968	7595	1434	1113
(Ochs and Brox, 2012)	468	1175	5863	1434	1595
(Lee et al., 2011)	288	34228	1785	64339	201
(Ma and Latecki, 2012)	189	806	1698	472	221
(Zhang and Shah, 2013)	155	**633**	1488	365	220
(Wang and Wang, 2014)	**151**	672	1121	359	204
(Wang et al., 2015)	209	796	1040	562	207
(Giordano et al., 2015)	278	824	**1029**	**192**	251
(Perez-Rua et al., 2015)	160	2708	3589	482	223
* (Wang et al., 2014a)	152	-	1053	-	**189**
* (Jain and Grauman, 2014)	189	1170	2883	333	228
ours w/o image saliency	226	1066	2404	309	348
ours with image saliency	233	1107	2383	319	346

Table 2.1: **Results on SegTrack v1.** *The entries show the average number of mislabelled pixels per frame (lower is better). Entries denoted by a dash '-' indicate that the authors did not evaluate their method on that particular video. Methods denoted by '*' are interactive.*

remarkable, given that our approach is simpler, does not require manual selection of the output segment, and is two orders of magnitude faster (sec. 2.4.4). The methods of (Ma and Latecki, 2012; Zhang and Shah, 2013) achieve lower errors than ours on average, but they are much slower (sec. 2.4.4).

If we look at more recent works than ours (Papazoglou and Ferrari, 2013), we see that the method of (Jain and Grauman, 2014) is on par with our method despite requiring a user to manually segment the first video frame. Similarly, our method is roughly on par with the interactive method of (Perez-Rua et al., 2015). The ranking object proposals method of (Wang and Wang, 2014) achieves lower errors than ours on average, but is much slower (sec. 2.4.4). Finally, the very recent methods of (Wang et al., 2015; Giordano et al., 2015), which follow our scheme, achieve lower errors on most videos.

Fig. 2.6 shows example frames from all 5 videos. Our method accurately segments all videos but *girl*, as it misses parts of her legs and arms. The higher error on *parachute* is due to including the paratrooper in the segmentation, as it is not annotated in the ground-truth. Note the high quality of the segmentation on *monkeydog* and *cheetah*, which feature fast camera motion and strong non-rigid deformations.

In general, inspecting the results reveals that most recent segmentation methods,

Figure 2.6: **Example results on SegTrack v1.** *We show 4 example frames per video, with the output of our method overlaid in green. Top to bottom: monkeydog, cheetah, birdfall, parachute, girl.*

precision	ours	(Yang et al., 2016)	(Faktor and Irani, 2014)	(Wang et al., 2014a)
birdfall	61.2	66.2	**74**	70.3
cheetah	36.5	-	**69**	-
girl	72.8	81.6	**91**	90.5
monkeydog	78.1	**78.7**	78	-
parachute	90.6	89.9	**94**	92.4
frog	77.0	80.7	**83**	81.1
worm	76.7	**81.6**	81	80.4
soldier	68.7	83.4	83	**85.3**
monkey	64.8	68.5	71	**89.8**
bird of paradise	92.2	**94.5**	-	**94.5**
bmx	62.8	-	**79**	-
drift	77.1	-	**86**	-
hummingbird	41.9	-	**75**	-

Table 2.2: **Results on SegTrack v2.** *The entries show the average intersections over union (IoU) (higher is better). Entries denoted by a dash '-' indicate that the authors did not evaluate their method on that particular video. Note that (Faktor and Irani, 2014) does not report decimal points.*

including ours, solve this dataset well. All methods lock on the object in all videos and accuracy differences between methods are due to finer localization of the object boundaries. When also taking into account that it contains only 5 very short videos, we believe this dataset is saturated.

2.4.2 SegTrack v2

SegTrack v2 (Li et al., 2013) adds 9 more videos (*drifting car, hummingbird, frog, worm, soldier, monkey, bird of paradise, BMX*) to the original 6 (sec. 2.4.1). The additional videos are of higher resolution and image quality and offer new challenges such as water reflections and smoke. This version of the dataset foreground also has pixel-level ground-truth for the foreground objects in every frame. In contrast to the first version however, if multiple foreground objects exist in a video, each is given a separate label. For the purposes of foreground object segmentation, where we only need to separate the foreground from the background (sec. 1.1), we consider the foreground segment to be the union of the all individual object segments in the video.

precision	ours	(Zhang et al., 2015)
birdfall	**226**	339
cheetah	1121	**803**
girl	2404	**1459**
monkeydog	**322**	365
parachute	348	**196**
frog	2301	**1272**
worm	**968**	1497
soldier	2054	**1879**
monkey	3586	**2526**
bird of paradise	2590	**1764**
drift	**7208**	-
bmx	**6686**	-
hummingbird	**17116**	-

Table 2.3: **Results on SegTrack v2.** *The entries show the average number of mislabelled pixels per frame (lower is better). Entries denoted by a dash '-' indicate that the authors did not evaluate their method on that particular video.*

Setup Several works that compare on SegTrack v2 use different measures to quantify segmentation performance. The two measures used more often are the average number of wrongly labeled pixels and the average intersection over union ratio of the segmentation and the ground-truth. To compare against other methods, we will adopt the measure used by the authors of each paper.

Note that this dataset was not available at the time of publication of our paper (Papazoglou and Ferrari, 2013). The work of (Faktor and Irani, 2014) was the first to evaluate on this dataset for video object segmentation and showed that our method achieved the state-of-the-art results on it until that point. All methods we compare to (Faktor and Irani, 2014; Wang et al., 2014a; Zhang et al., 2015; Perez-Rua et al., 2015; Wang et al., 2015; Yang et al., 2016) appeared after the publication of our paper (Papazoglou and Ferrari, 2013).

The method of (Wang et al., 2014a) is interactive: a user must manually mark a few frames that contain the foreground object. The method of (Zhang et al., 2015) is a class-specific segmentation method. It uses a detector, trained on manually annotated still images, to create a rough initial segmentation. The methods of (Faktor and Irani, 2014; Wang et al., 2015; Yang et al., 2016) are fully automatic. The methods of (Wang et al., 2015; Yang et al., 2016) follow a similar approach to ours, where they produce

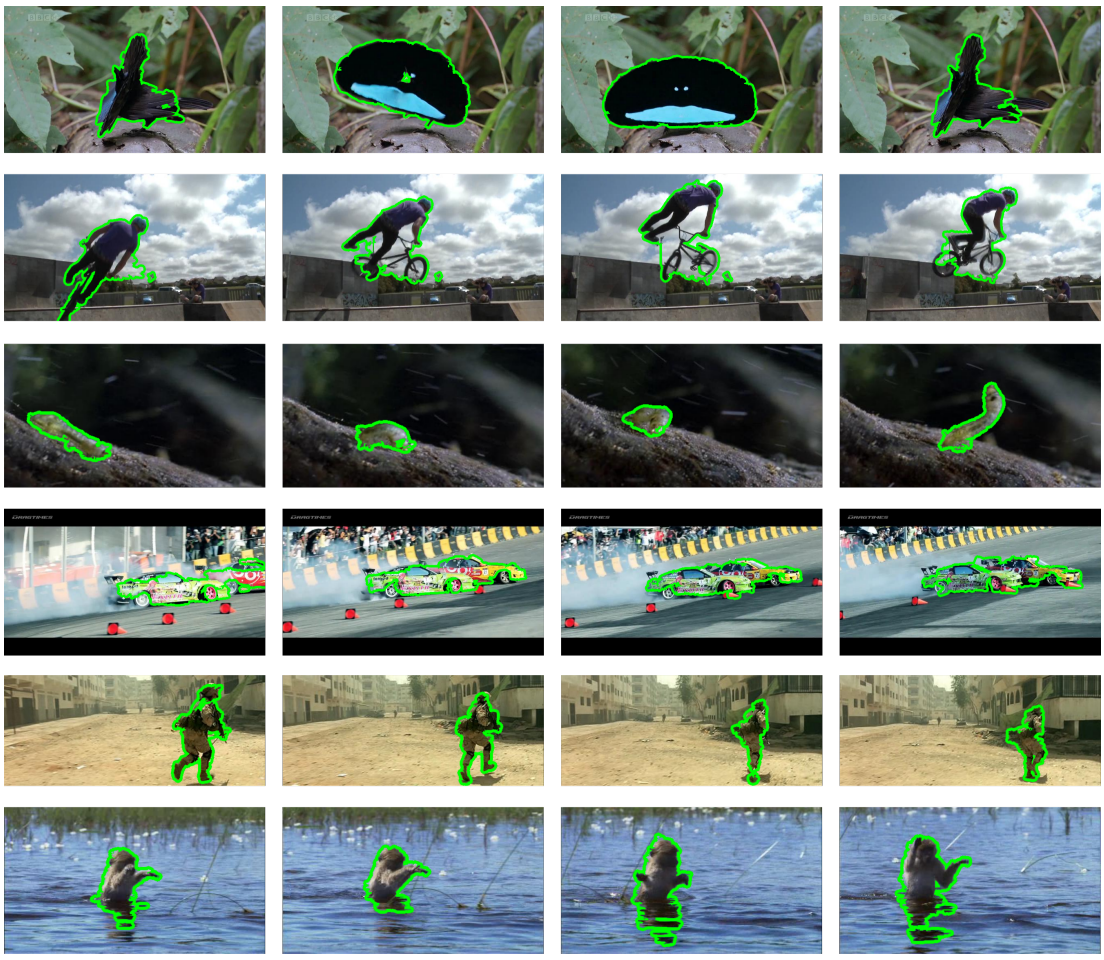

Figure 2.7: *Example results on SegTrack v2. We show 4 example frames per video, with the output of our method overlaid in green.*

an initial rough segmentation and then refine it using an MRF formulation.

Results. Our results are close to (Yang et al., 2016) for several videos, and slightly worse for the rest (table 2.2). The results of (Faktor and Irani, 2014) seem to be better on average. The method of (Wang et al., 2014a) has comparable or better results to us, but it does require manual segmentation of some frames. Compared to the class-specific method of (Zhang et al., 2015), we get better results in 3 videos out of 9. Figure 2.7 shows qualitative results for this dataset.

	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	avg
(Brox and Malik, 2010)	53.9	19.6	38.2	37.8	32.2	21.8	27.0	34.7	45.4	37.5	34.8
(Prest et al., 2012)	51.7	17.5	34.4	34.7	22.3	17.9	13.5	26.7	41.2	25.0	28.5
(Joulin et al., 2014)	25.1	33.3	27.8	34.1	42.0	28.4	35.7	35.6	22.0	25.0	30.9
(Stretcu and Leordeanu, 2015)	38.3	62.5	51.1	54.9	**64.3**	52.9	44.3	43.8	41.9	**45.8**	50.0
(Wang and Wang, 2016)	63.0	**69.0**	40.0	61.0	48.0	46.0	67.0	53.0	47.0	38.0	53.2
(Kwak et al., 2015)	56.5	66.4	**58.0**	**76.8**	39.9	**69.3**	50.4	56.3	53.0	31.0	**55.8**
ours w/o image saliency	**71.9**	64.4	39.6	64.3	54.5	37.8	68.8	56.3	52.0	26.7	53.6
ours with image saliency	71.2	67.3	40.3	64.3	56.0	44.1	**70.9**	**57.8**	**56.0**	23.3	55.1

Table 2.4: **Results on YouTube-Objects.** *The entries show the average per-class CorLoc ('aero' to 'train') as well as the average over all classes ('avg').*

2.4.3 YouTube-Objects v1

Dataset. YouTube-Objects v1 (Prest et al., 2012)[3] is a large database collected from YouTube containing many videos for each of 10 diverse object classes. The videos are completely unconstrained and very challenging, featuring large camera motion, diverse backgrounds, illumination changes and editing effects (e.g. fade-ins, flying logos). The objects undergo rapid movement, strong scale and viewpoint changes, non-rigid deformations, and are sometimes clipped by the image border (fig. 2.8). The dataset also provides ground-truth bounding-boxes on the object of interest in one frame for each of 1407 video shots.

Setup. We adopt the CorLoc performance measure of (Prest et al., 2012), i.e. the percentage of ground-truth bounding-boxes which are correctly localized up to the PASCAL criterion (Everingham et al., 2010) (intersection-over-union ≥ 0.5). For the purpose of this evaluation, we automatically fit a bounding-box to the largest connected component in the pixel-level segmentation output by our method. We evaluate two versions of our method, one that uses only the motion boundaries to produce the rougn initial segmentation and one that uses image saliency as an additional cue (sec. 2.3.3). We set the α weights for each version of the method using grid search on a separate video dataset that contains 22 videos depicting cars in racing sequences (sec. 4.6.1). We then use these weights for all 1407 shots in the YouTube-Objects dataset.

We compare to several recent methods (Prest et al., 2012; Joulin et al., 2014; Stretcu and Leordeanu, 2015; Wang and Wang, 2016; Kwak et al., 2015) that address the fore-

[3] `http://groups.inf.ed.ac.uk/calvin/learnfromvideo`

ground object segmentation setting, and report their performance as originally stated by the authors. Unfortunately, some works (Jain and Grauman, 2014; Rochan and Wang, 2014) only evaluate on a small subset of the dataset. As such, their published results are not directly comparable to ours.

Note that some of the works we compare to (Prest et al., 2012; Joulin et al., 2014; Kwak et al., 2015) are co-segmentation methods: they segment all the shots of an object class simultaneously. Co-segmentation methods take advantage of this setting to select segments that are similar in appearance between shots. In contrast, our method is only using information from each shot individually in order to build its appearance model. As these methods return a single foreground segment per shot, they are directly comparable to ours.

For (Brox and Malik, 2010), we report their performance as originally stated in (Prest et al., 2012). Since (Brox and Malik, 2010) solves the multi-object segmentation problem (chapter 1.1), they report the results for the segment with the maximum overlap with the ground-truth bounding-box for comparison (analogous to our experiment on SegTrack).

We also run (Lee et al., 2011) on 50 videos (5/class) using the implementation by their authors[4], as it is too slow to run on the whole database. For evaluation we fit a bounding-box to the top ranked output segment.

Results. On the 50-video subset, our method produces 42.0% CorLoc, considerably above the 28.0% reached by (Lee et al., 2011). This departs from what is observed on SegTrack v1 and suggests that our method generalizes better to a wide variety of videos.

Table 2.4 summarises our results on YouTube-Objects v1. We can see that using image saliency as an additional cue improves the segmentation accuracy on most object classes (with the exception of aeroplane and train). On average, using image saliency brings a moderate improvement from 53.6% to 55.1% CorLoc.

Our method outperforms the best segment produced by (Brox and Malik, 2010) (28.5%), confirming what we observed on the SegTrack v1 dataset. Moreover, our method performs substantially better than the co-segmentation method of (Prest et al., 2012). Interestingly, our method also performs significantly better than more recent co-segmentation methods (Joulin et al., 2014; Stretcu and Leordeanu, 2015) which were published after ours. The method of (Kwak et al., 2015) is on par with ours (55.8%).

[4] https://webspace.utexas.edu/yl3663/~ylee/

Figure 2.8: **Example results on YouTube-Objects v1.** *We show 4 example frames per video, with the output of our method overlaid in green.*

However, our method can segment each shot individually while co-segmentation methods require multiple shots from each object class in order to segment the foreground object.

Compared to the very recent method of (Wang and Wang, 2016), which segments each video individually like we do, we get comparable results (53.2%) if we only use motion boundaries. However, our method performs better when we add image saliency.

Figure 2.8 shows example results. The *cat*, *dog*, and *motorbike* examples show fast camera motion, large scale and viewpoint changes, and non-rigid deformations. On the bird video our method segments both the bird and the hand, as it considers them both foreground. The horse example shows that our method correctly segment objects even if largely clipped by the image border in some frames, as it automatically transfers object appearance learned in other frames. Videos of qualitative results are included in the supplementary material.

2.4.4 Runtime

Given optical flow and superpixels, our method takes **0.5 sec/frame** on SegTrack (0.05 sec for the inside-outside maps and the rest for the foreground-background labelling refinement). In contrast, (Lee et al., 2011) takes $>$ 300 sec/frame, with about 120 sec/frame for generating the object proposals (Endres and Hoiem, 2010). The point track clustering method (Brox and Malik, 2010) takes 7-44 sec/frame depending on the video, and (Ochs and Brox, 2012) takes 43-360 sec/frame. While (Ma and Latecki, 2012; Zhang and Shah, 2013; Wang and Wang, 2014, 2016) do not report timings nor have code available for us to measure, their runtime must be $>$ 120 sec/frame as they also use the object proposals of (Endres and Hoiem, 2010).

All timings were measured on the same computer (Intel Core i7 2.0GHz), and exclude optical flow computation which most methods use (with the exception of Giordano et al. (2015)). High quality optical flow can be computed rapidly on a GPU using (Sundaram et al., 2010) ($<$ 1 sec/frame). For superpixels we use SLIC (Achanta et al., 2012), which takes 0.005 sec/frame to compute on our CPU.

This analysis shows that at the time of publishing our method (Papazoglou and Ferrari, 2013) was a lot faster than existing competitors and was in fact efficient enough to be applied to very large collections of videos. The later method of (Wang and Wang, 2014) which relies on object proposals is still orders of magnitude slower than ours due to their reliance on computationally expensive object proposals. Instead, the recent method of (Giordano et al., 2015) achieves even faster computation as they do not rely on optical flow to produce the initial rough segmentation.

Chapter 3

Class-specific video object segmentation

3.1 Introduction

In the previous chapter we focused on class-agnostic segmentation, where the object class of that the foreground objects belongs to is not known a-priori. Lacking any prior knowledge about the appearance of the foreground objects, we relied mainly on motion saliency: we assumed that the foreground object moves differently to the background in at least some part of the video. This approach leads to a generic method that can be applied to any video, regardless of what class the foreground objects belong to.

However, in many applications the class of the foreground objects is known in advance. For instance, one could search for cars in YouTube and download the top 100 results (Prest et al., 2012). We do not know in which frames the cars appear, their location in these frames or their size, but we do know that the objects that we are interested in are cars. This allows us to add prior knowledge about the appearance of the class to our segmentation method, as an additional cue to complement motion saliency.

In this chapter we present an extension of the class-agnostic method presented in chapter 2. The key idea is that we can use modern object class detectors (Girshick et al., 2014) in order to detect the objects in individual frames (fig. 3.1). We can then use these detections as additional cues in our segmentation formulation in order to produce a more accurate result.

Similar to the motion saliency cue, the object detector output alone can only be considered a rough estimate of the object's location. The object detector does not label

Figure 3.1: **Example of a bounding-box produced by a modern object detector (Girshick et al., 2014).** *(Left) Original frame. (Right) The detection with the highest score for the car class.*

individual pixels as belonging to the object or not, but marks the object by a bounding-box around it. Furthermore, this bounding-box does not always tightly contain the object (fig 3.1). Additionally, the object detector output is typically sparse in time (it detects the object in only a few frames) and can produce false positives. However, this rough estimate can be refined using our technique (sec. 2.3.2), to produce a pixel-level segmentation of the object that is both more accurate and dense than the object detector output alone.

This chapter contains new material that is yet unpublished.

3.2 Related work

Our approach is most related to tracking-by-detection methods (*e.g.* Huang and Nevatia (2010); Andriyenko and Schindler (2011); Santhoshkumar et al. (2013); Akin and Mikolajczyk (2014); Kuo et al. (2009); Li et al. (2009); Yang and Nevatia (2014)). Tracking-by-detection approaches apply a class-specific object detector in every frame individually, and then associate the detection into tracks (one track per object). The common approach to solve the association problem, is to integrate several cues such as appearance, motion, detection size and location into an affinity model to measure the similarity between the detections and then optimise an association model. Over the years, a lot of works have proposed several different affinity models (Kuo et al., 2009; Li et al., 2009; Yang and Nevatia, 2014) and association models (Huang and Nevatia, 2010; Andriyenko and Schindler, 2011; Santhoshkumar et al., 2013; Akin and Mikolajczyk, 2014).

As we already noted, the output bounding-boxes produced by object detectors are typically sparse and may not be tight around the objects. Due to that, just associat-

ing the detections is not sufficient to produce accurate tracking in every frame. In order to overcome these challenges, various techniques complement the association step. A popular approach to overcome these challenges, is training a supplementary, instance-specific detector, using the original detections as training data. Another complementary technique to fill in the gaps in some frames due to missed detections, is interpolating the bounding-boxes of past and future detections.

Also related to our approach is the work of (Zhang et al., 2015). Similar to us, the authors use an object detector to generate a set of detections (bounding-boxes) for each frame. These detections are then supplemented with class agnostic object proposals to form a set of possible object locations for each frame. The authors then use point trajectories to track individual pixels through time. They then select a set of object trajectories that pass through multiple high-scoring object proposals and detections. A final pixel-level segmentation is produced by refining the selected set of trajectories, using shape priors. Compared to this approach, our method is relatively simpler: we just convert the per-frame bounding-box-level detections into a single pixel-level heatmap indicating the likelihood of a particular pixel to belong to the foreground. We can then use that directly as an additional unary potential in our energy formulation to produce a final segmentation (sec. 3.3).

3.3 Segmentation model with class-specific appearance term

We follow the same generic segmentation model presented for the class-agnostic case (eq. 2.6), which is a sum of unary and pairwise terms. In the case of class-specific segmentation we can use an additional class-appearance unary term, C which evaluates how likely a superpixel is to belong to the foreground (fig. 3.3):

$$U(\mathcal{L}) = \sum_{t,i} A_i^t(l_i^t) + \alpha_L \sum_{t,i} L_i^t(l_i^t) + \alpha_C \sum_{t,i} C_i^t(l_i^t) \tag{3.1}$$

As a reminder, $l_i^t \in \{0,1\}$ denotes the label of superpixel s_i^t in frame t. A labelling $\mathcal{L} = \{l_i^t\}_{t,i}$ of all superpixels in all frames represents a segmentation of the video. The terms A and L correspond to the appearance and location model respectively (sec. 2.3.2).

We now present how we compute the class appearance term C. To main idea is to use RCNN (Girshick et al., 2014), a modern object class detector to score how

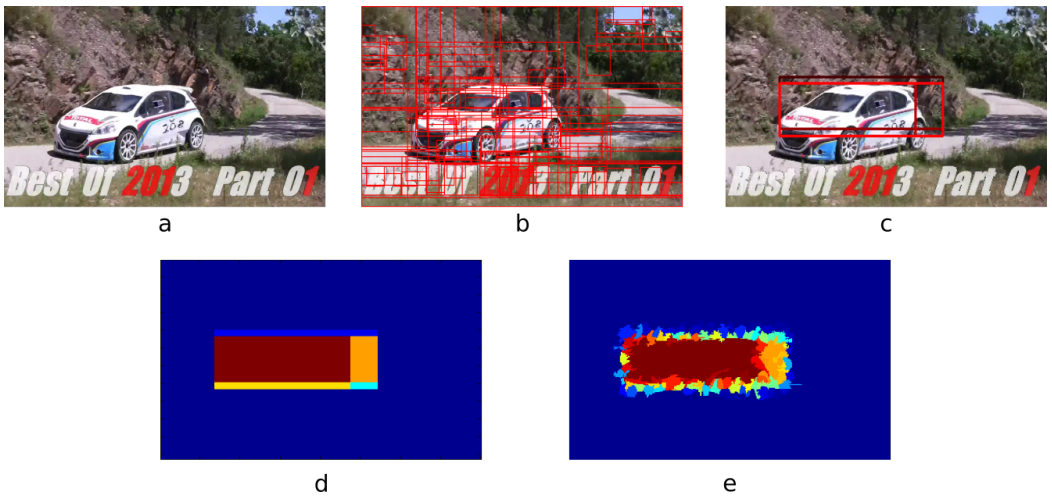

Figure 3.2: ***Class-specific term.*** *This figure visualises the computation of the class-specific term C.* ***(a)*** *The original video frame,* ***(b)*** *the Selective Search object proposals (Uijlings et al., 2013).* ***(c)*** *The object proposals with a positive score according to the RCNN detector (Girshick et al., 2014). Brighter red colour corresponds to higher scores.* ***(d)*** *A heatmap visualisation of the pixel scores C_p.* ***(e)*** *A heatmap visualisation of the superpixel scores C_s.*

likely each superpixel is to belong to the foreground class. RCNN uses the AlexNet network architecture (Krizhevsky et al., 2012), and is trained on the PASCAL VOC 2012 dataset (Everingham et al., 2012).

Like all modern object detectors, RCNN outputs a set of bounding-boxes (fig. 3.2) and corresponding confidence scores for each bounding-box. Since the scores correspond to specific bounding-boxes, many of which are intersecting, they are not readily available to score individual pixels (and by extension, superpixels).

In order to score each individual superpixel, we use the following procedure. For each video frame, we extract a set of class-agnostic object proposals (fig. 3.2b) using Selective Search (Uijlings et al., 2013). We then score each object proposal using the RCNN object detector (Girshick et al., 2014), and keep the object proposals o that have a positive score C_o for the class that we are interested in (fig. 3.2c). The score of each individual pixel p is then (fig. 3.2d):

$$C_p = \sum_{\{o \ni p | C_o > 0\}} C_o \tag{3.2}$$

where $o \ni p$ denotes that the object proposal o contains pixel p.

The score C_s of a superpixel s is then the average score of its pixels (fig. 3.2e):

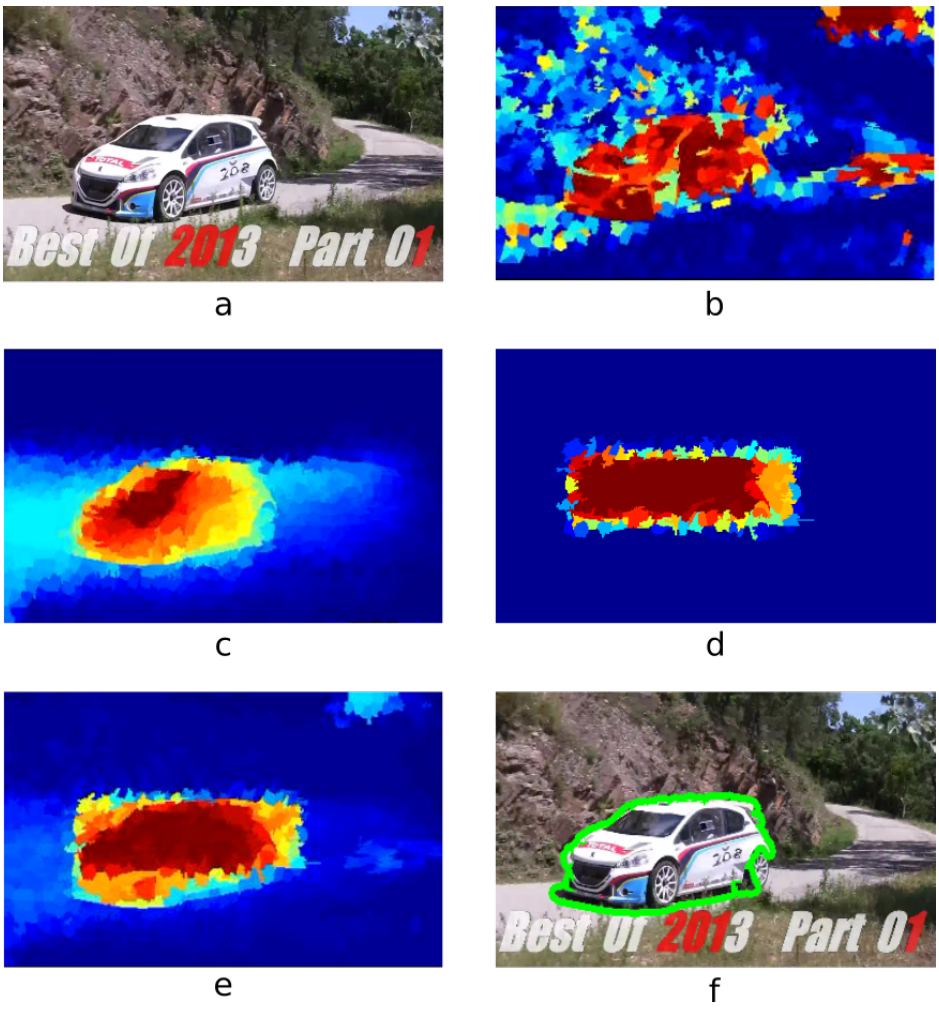

Figure 3.3: ***Unary potentials.*** *This figure shows a heatmap visualisation of the individual unaries (eq. 3.1).* ***(a)*** *The original video frame.* ***(b)*** *The appearance likelihood A.* ***(c)*** *The location model L.* ***(d)*** *The class-appearance term C.* ***(e)*** *The combined unary term U.* ***(f)*** *The resulting segmentation.*

$$C_s = \frac{1}{|s|} \sum_{p \in s} C_p \tag{3.3}$$

where $|s|$ denotes the number of pixels in s.

3.4 Experimental evaluation

We evaluate our method on the YouTube-Objects dataset, which is described in detail in section 2.4.3. YouTube-Objects contains 10 object classes which are a subset of the 20 classes in the PASCAL VOC dataset (Everingham et al., 2012). This allows us

Figure 3.4: ***Tang et al. (2013) annotation examples.*** *A few examples of the segmentation annotations provided by (Tang et al., 2013). These annotations were collected in a semi-automated manner and are often erroneous.*

to use the RCNN detector as released in (Girshick et al., 2014) as it is trained on the PASCAL VOC.

Setup. Similarly to section 2.4.3, we adopt the CorLoc performance measure of (Prest et al., 2012). We set the α weights using grid search of a separate video dataset containing 22 videos of cars in racing sequences.

We compare our method to two unsupervised methods (Stretcu and Leordeanu, 2015; Wang and Wang, 2016) as well as three co-segmentation methods (Prest et al., 2012; Joulin et al., 2014; Kwak et al., 2015). For information on these methods we refer the reader to sections 2.2 and 2.4.3.

As a first baseline, we also compare against our unsupervised method (chapter 2). As a second baseline, we compare against the standalone RCNN detector (Girshick et al., 2014). This allows us to evaluate whether our model significantly contributes to the segmentation performance, or the performance can be attributed mostly on the RCNN detector. For evaluation, in each frame we keep the detection with the highest score for the class shown in the video. Note, that we only evaluate on frames that do contain the class.

Finally, we compare against a classical tracking-by-detection method (Marin-Jimenez et al., 2014). This tracker casts the detection association problem as a grouping problem. It uses three similarity measures between pairs of detections based on their appearance, their location and the number of point tracks that pass through both of them. Then they use the Clique Partitioning algorithm of Ferrari et al. (2001) to group them. Finally, it uses linear interpolation to fill in the gaps in frames with missed detections. This method is representative of a typical tracking-by-detection technique such as (Huang and Nevatia, 2010; Andriyenko and Schindler, 2011; Santhoshkumar et al., 2013; Akin and Mikolajczyk, 2014). Note that as input, we use the same object de-

	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	avg
(Prest et al., 2012)	51.7	17.5	34.4	34.7	22.3	17.9	13.5	26.7	41.2	25.0	28.5
(Joulin et al., 2014)	25.1	33.3	27.8	34.1	42.0	28.4	35.7	35.6	22.0	25.0	30.9
(Kwak et al., 2015)	56.5	66.4	58.0	**76.8**	39.9	**69.3**	50.4	56.3	53.0	31.0	55.8
(Stretcu and Leordeanu, 2015)	38.3	62.5	51.1	54.9	64.3	52.9	44.3	43.8	41.9	**45.8**	50.0
(Wang and Wang, 2016)	63.0	69.0	40.0	61.0	48.0	46.0	67.0	53.0	47.0	38.0	53.2
RCNN (Girshick et al., 2014)	42.2	47.1	28.1	64.3	20.9	48.0	39.7	37.5	30.0	13.8	37.2
(Marin-Jimenez et al., 2014)	52.0	52.9	31.6	38.8	**61.6**	59.8	58.2	56.2	47.0	44.0	50.2
ours class-agnostic	71.2	67.3	40.3	64.3	56.0	44.1	70.9	57.8	56.0	23.3	55.1
ours class-specific	**78.4**	**71.2**	**59.7**	75.0	44.0	56.7	**76.6**	**61.7**	**65.0**	40.5	**62.9**

Table 3.1: **Results on YouTube-Objects.** *The entries show the average per-class CorLoc ('aero' to 'train') as well as the average over all classes ('avg').*

tections used by our method (RCNN) as opposed to the original publication (Marin-Jimenez et al., 2014) which used the weaker Deformable Part Models (Felzenszwalb et al., 2010). Like all tracking-by-detection algorithms, the method groups the detections into tracks.

Selecting a track for evaluation is not trivial as there are multiple tracks passing through every frame. Furthermore, each track is typically shorter than the length of the shot, so more than one are needed to produce a segmentation for every frame. In our experiments, we rank the tracks according to the average score of their detections. Then, for each frame we select the highest ranked track that passes through it.

To the best of our knowledge, the work of (Zhang et al., 2015) is the only other class-specific video object segmentation method. However, it only evaluates on a subset of the YouTube-Object dataset. Furthermore, for ground-truth data, they use the pixel-level annotations of (Tang et al., 2013) which were collected in a semi-automated manner and are often erroneous (fig. 3.4). As such, we do not believe that they should be used for performance evaluation. Due to the lack of freely available code for the method, we could not compare against it.

Results. As shown in table 3.1, adding class-specific knowledge significantly increases the performance of our method compared to the class-agnostic version (sec. 2.3), for all object classes except cat. This is expected, as class appearance is a strong cue. Our class-specific method is also able to significantly outperform all unsupervised (Stretcu and Leordeanu, 2015; Wang and Wang, 2016) and co-segmentation methods (Prest et al., 2012; Joulin et al., 2014; Kwak et al., 2015). The poor per-

formance on the cat class can be attributed to the particularly low precision of the classifier on that class (28.1%) which can mislead our method. This is most likely due to many videos showing the animals in poses and viewpoints that were not seen in the training set (*e.g.* laying on their back).

Interestingly, our method performs much better than the standalone detector (Girshick et al., 2014). This indicates that the performance improvement over our unsupervised method is not simply attributed to the detector being very accurate.

The tracking-by-detection algorithm (Marin-Jimenez et al., 2014) brings a considerable improvement compared to the single frame detector. By aggregating detections from multiple frames the tracking-by-detection algorithm has a more robust selection criterion. Due to that, a low scored detection on the object can be associated to higher scored detections in other frames and selected instead of a false positive that individually scores higher. Furthermore, the linear interpolation is able to fill in the gaps in frames with missed detections. Interestingly, our class specific segmentation technique performs even better. This performance difference is even more impressive if we consider that our method outputs a detailed pixel-level segmentation whereas the detector needs only produce a bounding-box around the object.

Figures 3.5 and 3.6 show qualitative results of our class-specific method compared to the class-agnostic method of chapter 2. We can see that adding class-specific appearance knowledge can help in cases where the object is undersegmented. This is usually due to limited motion which is not sufficient to produce clear motion boundaries. Furthermore, it can help disambiguate between the objects of interest and other moving objects in the foreground (*e.g.* dog and skateboard, or long shadows). Videos of qualitative results are included in the supplementary material.

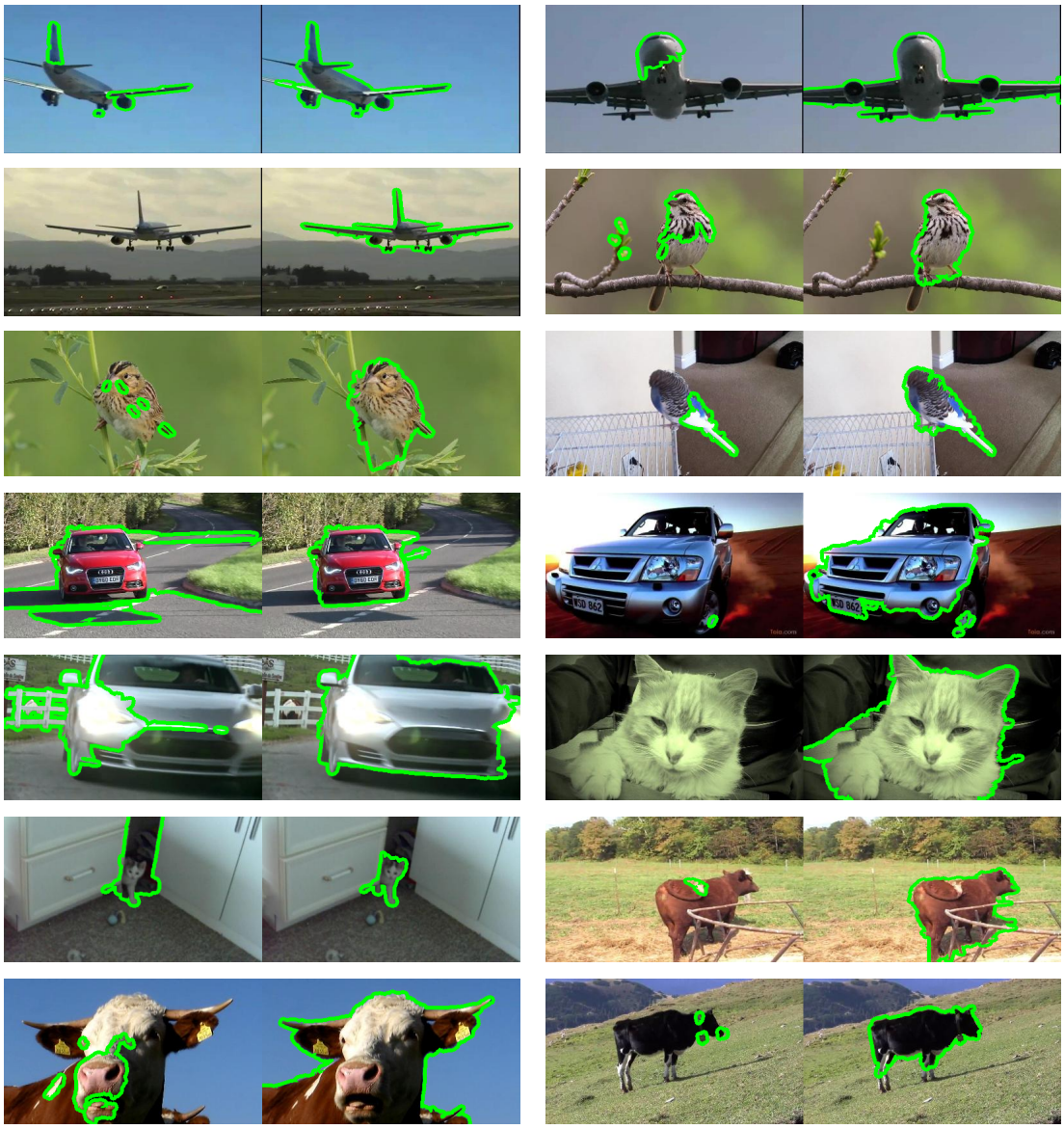

Figure 3.5: ***Qualitative results.*** *This figure shows a qualitative comparison of the class-agnostic segmentation method of chapter 2 (left) against the class specific segmentation (right) on the YouTube-Objects v1 dataset. This figure shows results for the aeroplane, bird, car, cat and cow classes.*

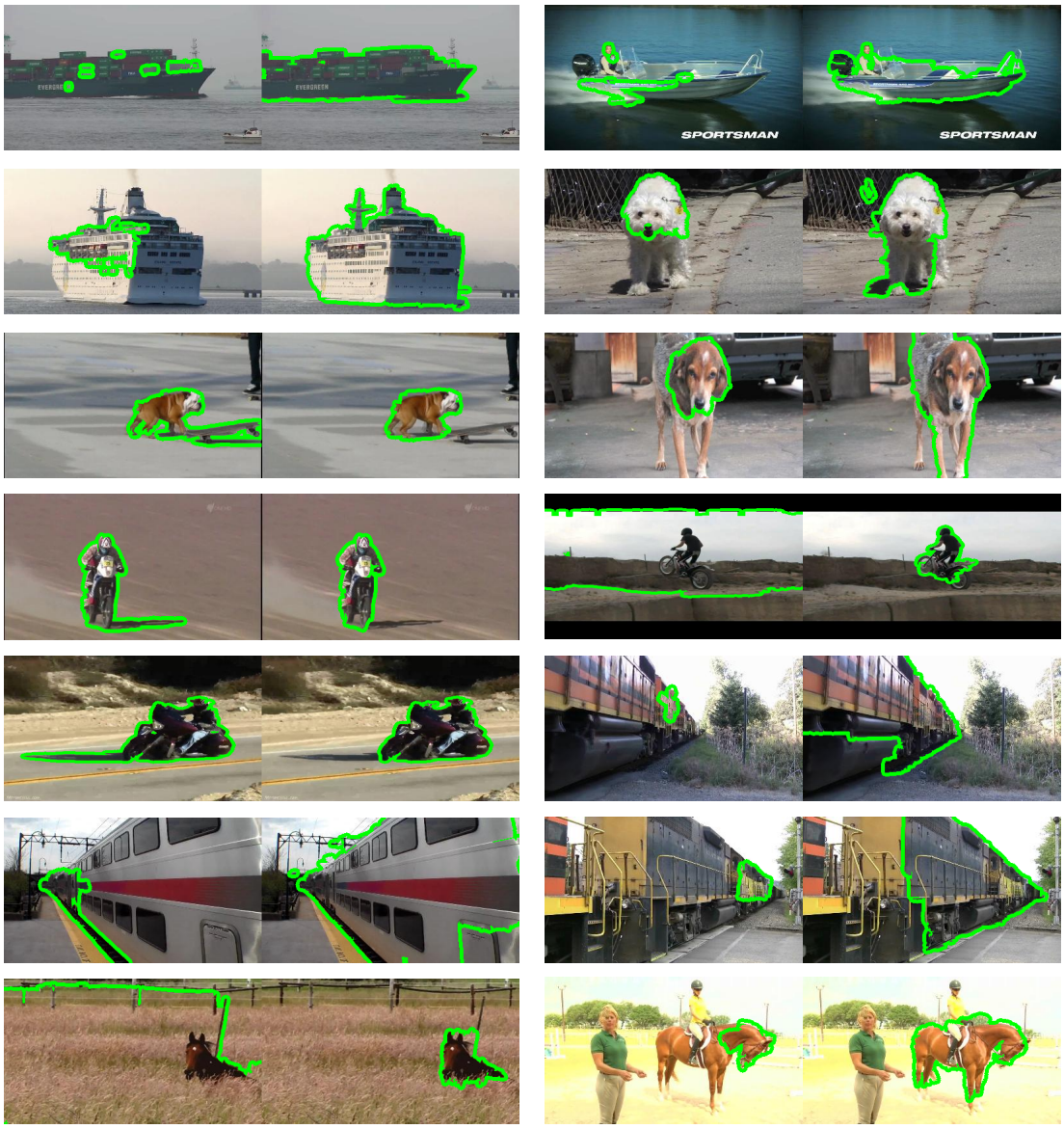

Figure 3.6: ***Qualitative results.*** *This figure shows a qualitative comparison of the class-agnostic segmentation method of chapter 2 (left) against the class specific segmentation (right) on the YouTube-Objects v1 dataset. This figure shows results for the boat, dog, motorbike, train and horse classes.*

Chapter 4

Temporal alignment of videos based on object viewpoint

4.1 Introduction

In this chapter we utilise video object segmentation to temporally align videos. Temporal alignment of videos is often a key step in several popular tasks, such as video morphing (Liao et al., 2014), video mosaicking and stitching (Agarwala et al., 2005), video compositing (Ruegg et al., 2013), video summarisation (Ngo et al., 2005), action recognition and video retrieval (Jiang et al., 2007) and High Dynamic Range (HDR) video (Kang et al., 2007). Much previous work on temporal alignment focuses on videos of the same scene recorded from multiple cameras (Caspi and Irani, 2000, 2002, 2001; Caspi et al., 2006; Wolf. and Zomet, 2006; Tuytelaars and van Gool, 2004; Evangelidis and Bauckhage, 2013; Wang et al., 2014b). Instead, we want to align videos that are only weakly related: we simply require that their main object belongs to the same semantic class. For example, two videos of different cars driving along different tracks, and against different backgrounds.

Our alignment method establishes frame-to-frame correspondences such that the two cars are seen from a similar viewpoint (e.g. facing right), while also being temporally smooth and pleasing to the eye (fig. 4.1). Our key intuition is that the object viewpoint is a good indicator of whether videos showing different instances of the same class are aligned correctly. If the aligned frames consistently display very similar viewpoint over time, the alignment looks accurate and visually pleasing (smooth frame transitions and without stuttering).

Possible applications are in computer graphics and special effects, where one could

replace an object in a video with another object of the same class from another video. This can be useful for placing ads in video search engines. For example, we could align a user video to a video ad, and blend a portion of the ad into it. It could even be used in combination with retrieval: a retrieval system could find the most suitable video from a large pool of ads, and then our method could produce the smooth temporal alignment necessary to blend in the ad. Another example is filming a stunt scene with a cheap car and then replace it with a more expensive one without fear of damaging it. Temporal re-ordering is necessary in both examples above, as we cannot expect the two videos to show events in the same order.

A few previous works (Rao et al., 2003; Ukrainitz and Irani, 2006; Dexter et al., 2009) have tackled aligning semantically similar videos. However, they typically assume that the videos show a scripted sequence of events (e.g. drinking motion (Dexter et al., 2009), hand waving (Rao et al., 2003)), possibly out of phase (*i.e.* the events occur at different, varying speeds). Under this assumption, finding an optimal alignment can be solved using the Dynamic Time Warping algorithm (DTW) (Sakoe and Chiba, 1978) (as in (Dexter et al., 2009; Rao et al., 2003)). However, this assumption is unrealistic for most real-world videos, where events may occur in a different order, or occur only in one of the videos.

Here, we present a method that is able to cope with such challenging videos. Our assumption is that we can decompose videos into contiguous temporal segments, and put them into correspondence so that each pair of corresponding segments (rather than the entire videos) show the same sequence of events (fig. 4.3). The main contribution of our approach is to solve the temporal segmentation and the correspondence problems jointly. For this, we use a principled probabilistic model defined over the space of all possible temporal segmentations and correspondences (sec. 4.3). A likelihood function promotes putting in correspondence segments showing similar viewpoints, while other components favour temporal consistency and smoothness. Inference in our model is a computationally intractable combinatorial problem. Therefore, we present a Markov Chain Monte Carlo (MCMC) sampling (Neal, 1993) procedure to search its complex parameter space efficiently (sec. 4.4).

We test our method on a set of 22 videos of cars racing in rally competitions collected from the internet, where we have manually annotated the viewpoint in each frame (sec. 4.6.1). These videos are challenging, showing fast motion, complex and changing backgrounds as well as different car models. We automatically split them into different shots using (Kim and Kim, 2009), but they are otherwise untrimmed

Figure 4.1: ***Viewpoint-driven temporal alignment.*** *The goal of this task is to align the two videos so that both of them show the object from the same viewpoint frame-by-frame as shown above. This example alignment was produced by our method.*

and unedited. This is different from videos used in previous work (Rao et al., 2003; Ukrainitz and Irani, 2006; Dexter et al., 2009), which are trimmed so that they show the exact same sequence of events in their entirety.

In our videos, events are often in a different order and occur a different number of times. Hence, determining their optimal alignment can be ambiguous, i.e. we cannot define a unique ground-truth alignment as in (Dexter et al., 2009). For instance, if a certain viewpoint appears only once in a video and multiple times in the other, there are multiple valid ways of aligning them (e.g. fig. 4.2). To address this, we perform a comprehensive evaluation that takes into account several different factors: viewpoint similarity, temporal consistency and visual pleasingness. We evaluate these factors quantitatively on our dataset using a new carefully designed evaluation protocol, and with a substantial user study on visual pleasingness (this is in contrast to previous work that are mostly evaluated qualitatively on a few videos, *e.g.* (Rao et al., 2003; Ukrainitz and Irani, 2006)). Our results show that our method is superior to three alternative alignment methods (sec. 4.6.2), including the popular DTW (Sakoe and Chiba, 1978).

This work has been accepted for publication in the Asian Conference on Computer Vision (ACCV) 2016.

4.2 Related work

Previous works on temporal alignment can be categorised based on their assumptions about the input videos.

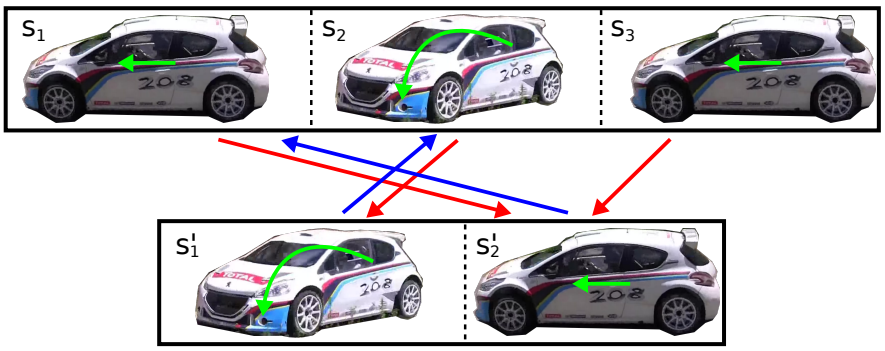

Figure 4.2: *Example alignment between sequences of events that cannot be aligned just by stretching and shrinking the time domain of the videos.*

Videos of the same scene from different views. Most previous works, e.g. (Caspi and Irani, 2000, 2002, 2001; Caspi et al., 2006; Wolf. and Zomet, 2006; Tuytelaars and van Gool, 2004) focus on joint spatio-temporal alignment of videos of the same dynamic scene, recorded by two uncalibrated cameras placed at different viewpoints (typically stationary). (F. L. C. Padua, 2009) also attempts to spatio-temporally align videos of a single dynamic scene, but they jointly process videos from multiple cameras instead of just two. The work of (Douze et al., 2016) also assumes a single dynamic scene recorded by multiple cameras, but focuses on temporal alignment only.

These works assume a fixed but unknown geometric transformation (homography) along the entire sequence. The videos are also assumed to have a linear temporal relationship (a constant time offset and possibly different frame rates). There are two general approaches explored. The first (Caspi and Irani, 2000, 2002, 2001; Wolf. and Zomet, 2006; Tuytelaars and van Gool, 2004) is to extract feature trajectories in each video. These trajectories are then typically matched between videos using a RANSAC procedure to produce a spatio-temporal alignment. The second approach (Caspi and Irani, 2000, 2002, 2001; Wolf. and Zomet, 2006) relies directly on image intensities to estimate a homography and an affine temporal alignment.

Videos of the same scene at different times. A few works (Diego et al., 2013; Evangelidis and Bauckhage, 2013; Wang et al., 2014b) focus on spatio-temporal alignment of videos of the same scene, but taken at a different time. To compensate for the lack of temporal overlap between the input videos, these works assume the cameras follows roughly the same trajectory.

Similar to the previous category of works Diego et al. (2013) uses a combination

of feature trajectories and image intensities and formulate a spatio-temporal alignment energy model to jointly estimate the temporal affine transformation and homography. The model is then optimised following an alternating optimisation approach. In (Evangelidis and Bauckhage, 2013) the authors compute per-frame interest points which they hash into short descriptors. Temporal alignment is posed as a voting scheme, where each frame votes for possible correspondences based on the descriptor similarity. In (Wang et al., 2014b) the authors aim for interactive video alignment and use a modified DTW approach with additional constraints (*i.e.*, some frames are manually matched).

Videos of semantically similar scenes. Our work falls in the category of temporal alignment of videos that do not show the same scene, but rather semantically similar content (e.g. drinking motion (Dexter et al., 2009), hand waving (Rao et al., 2003)). Typically, the videos depict people performing some scripted sequence of actions, such as drinking or waving (Rao et al., 2003; Ukrainitz and Irani, 2006; Dexter et al., 2009), and the goal is putting in correspondence frames showing the same body pose. These approaches typically align short videos showing the exact same sequence of events (possibly at different speed) and cannot handle challenging videos showing events in a different order like we do.

In (Rao et al., 2003) the authors use 3D point trajectories to create an affinity matrix and align the sequences using DTW. In (Ukrainitz and Irani, 2006) the authors formulate temporal alignment as maximising local space-time correlations. Their algorithm is based on direct image intensities and assumes videos with static cameras, simple backgrounds and similarly sized objects. In the work of (Dexter et al., 2009) the authors propose temporal descriptors based on point trajectories. The videos are then aligned using DTW.

4.3 Temporal alignment model

Our goal is to align two videos where different events may appear in a different order. Fig. 4.2 shows a simple example, featuring two types of events: going straight (s_1, s_3, s'_2) and turning left (s_2, s'_1). Ideally, we would like to match s'_1 to s_2 and s'_2 to either s_1 or s_3 (both would be valid). Note how the problem is not symmetric: when aligning the second video to the first, we would like to align s_1 to s'_2, s_2 to s'_1, and s_3 to s'_2 again. The events can have a different duration, but they can still be matched by

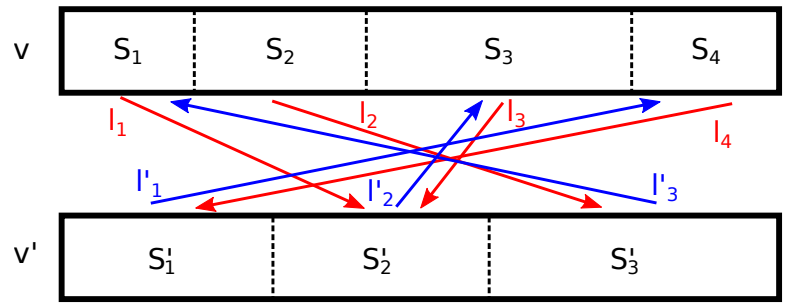

Figure 4.3: *One possible configuration of temporal segmentations S, S' and correspondences l between two videos v, v'. Each video is partitioned into a series of contiguous temporal segments: $S = \{s_1, s_2, s_3, s_4\}$ and $S' = \{s'_1, s'_2, s'_3\}$. Each segment has a correspondence in the other video, denoted by l (arrows). Note that the correspondences are not necessarily mutual (e.g. $s_2 \rightarrow s'_3$, but $s'_3 \rightarrow s_1$).*

playing the shorter event slower to match the duration of the longer one. As such, we can deal with videos of different durations. Aligning this example requires shuffling the temporal order of the videos, and re-using some of its segments.

An additional challenge is that the temporal segmentation of the videos into different events is also not known in advance. Our method solves the temporal segmentation and segment correspondence problems jointly, using a single probabilistic model over the two tasks, which we now define formally.

Let v and v' be the two videos we want to align. $S = \{s_1, ..., s_N\}$ is the set of *contiguous* temporal segments composing v. The temporal segmentation S' of v' is defined analogously. The correspondence l_i indicates which segment from v' is matched to the i-th segment in v (l'_j is defined analogously; note that $l_i = j \implies l'_j = i$). The model parameters Θ include the temporal segments of both videos $\mathcal{S} = \{S, S'\}$ and the set $\mathcal{L}$ containing all correspondences (fig. 4.3). Note how both the segmentations S, S' and the correspondence $\mathcal{L}$ have a variable number of elements, as the number of segments in each video is not predefined. It is another parameter to be searched over during inference.

We define the posterior distribution over the parameters to be

$$p(\mathcal{L}, \mathcal{S}|D) = p(\mathcal{L}|\mathcal{S}, D) \cdot p(\mathcal{S}|D) \tag{4.1}$$

where D are appearance descriptors extracted for all frames in the videos. Since we want to align the videos so that they show the same viewpoint, we use state-of-the-art CNN descriptors (Girshick et al., 2014) which we specifically fine-tuned to classify different viewpoints (sec. 4.5). The two factors in the posterior compete to allow

our model to find alignments which put similar viewpoints in correspondence, while also being temporally smooth and visually pleasing. The *correspondence likelihood* $p(\mathcal{L}|\mathcal{S},D)$ promotes putting into correspondence temporal segments (across videos) that are consistently similar in appearance through time. The *temporal segmentation likelihood* $p(\mathcal{S}|D)$ promotes that each temporal segment is homogeneous in appearance (within a video). A homogeneous segment is likely to contain a single viewpoint, which makes it a good unit for matching across videos. Furthermore, it promotes having few temporal segments. Having too many segments can cause the alignment to look jerky due to the frequent segment switches over time, which is not visually pleasing. We now discuss each factor in more detail.

Correspondence likelihood. We define the correspondence likelihood to be

$$p(\mathcal{L}|\mathcal{S},D) = \prod_i p(l_i|\mathcal{S},D) \cdot \prod_j p(l'_j|\mathcal{S},D) \tag{4.2}$$

where each $p(l_i = k|\mathcal{S},D)$ evaluates the likelihood of $l_i = k$ according to the appearance similarity of s_i and s'_k (these factors are conditionally independent). We define the probability of one correspondence l_i to be

$$p(l_i = s'_j|\mathcal{S},D) \propto \exp\left(-\alpha_M \frac{||s_i||}{||v||} d(s_i,s'_j)\right) \tag{4.3}$$

where α_M is a scalar weight, $||s_i||$ is the length of segment s_i (*i.e.*, the number of frames in it), $||v||$ is the length of video v, and $d(s_i,s'_j)$ denotes the appearance distance between the segments s_i and s'_j.

We designed d so that it can evaluate whether the appearance of the segments is consistently similar through time. Since the segments can have different length (*i.e.*, different speed), we first put their frames in one-to-one correspondence, denoted by $(f \to f')$ (see fig. 4.4). We can now compute

$$d(s_i,s'_j) = \frac{\sum_{f\to f'} \mathrm{a}(f,f')}{\max(||s_i||,||s'_j||)} \tag{4.4}$$

where $\mathrm{a}(f,f')$ denotes the appearance distance between frames f and f' (sec. 4.5). Note that DTW (Sakoe and Chiba, 1978) is a reasonable alternative segment distance, as it also measures similarity through time. However, we found that d produces comparable results to DTW, while being computationally more efficient.

Temporal segmentation likelihood. The temporal segmentation likelihood $p(\mathcal{S}|D)$ promotes having a small number of segments that are homogeneous in terms of appearance.

$$p(\mathcal{S}|D) \propto \exp\left(-\alpha_T \sum_i \frac{||s_i||}{||v||}\Delta_i\right) \exp\left(-\alpha_P(||S||^2 + ||S'||^2)\right) \tag{4.5}$$

where α_T, α_P are scalar weights, Δ_i is the appearance distance a averaged over all pairs of frames within s_i (sec. 4.5, and $||S||$ and $||S'||$ are the number of segments in v and v', respectively. Note that we can compute Δ_i in constant time by using summed area tables (C., 1984). The ratio $\frac{||s_i||}{||v||}$ ensures that the contribution of each temporal segment is proportional to its length.

Note that the temporal segmentation likelihood is comprised by two factors. The first factor promotes segments that are homogeneous in terms of appearance. The second factor acts as a prior and promotes having a small number of segments. These two factors and the correspondence likelihood compete in order to strike a balance on the optimal number of segments.

On one hand, having many short segments results in a high $p(\mathcal{S}|D)$, which is trivially maximised when each frame forms its own segment (as then it is maximally homogeneous in appearance). In this limit case, $p(\mathcal{L}|\mathcal{S},D)$ reduces to a nearest-neighbour matching between individual frames in the two videos, which results in a low average appearance distance, but is also sensitive to noisy appearance descriptors. On the other hand, having a few long segments brings temporal smoothness and produces a more visually pleasing alignment.

4.4 Inference

Inferring the most likely Θ^* according to our model is a hard combinatorial problem, since we jointly optimise over the number of segments N,N', the position of their delimiters S,S', as well as the set of correspondences $\mathcal{L}$. Furthermore, the posterior (4.1) is a complex distribution which we cannot evaluate analytically. Thus, we use Markov Chain Monte Carlo (MCMC) sampling (Neal, 1993) to search the parameter space.

Following the standard formulation, at each iteration we propose a new sample Θ' from the current sample Θ using a proposal distribution $q(\Theta'|\Theta)$. Θ' is then accepted

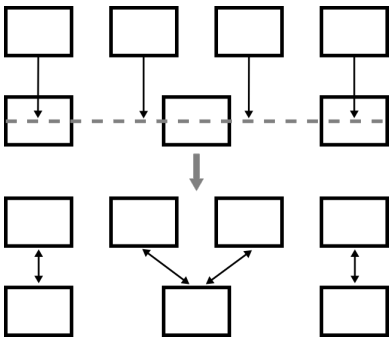

Figure 4.4: *Our appearance distance d (4.4) measures the similarity in appearance between two segments (sec. 4.3). We first put the segment frames in one-to-one correspondence. For this, we project the longest segment onto the shorter one (top), and put each frame in the longest segment in correspondence with the frame closest to the projection (bottom). d is the distance in appearance averaged over all corresponding frames (4.4).*

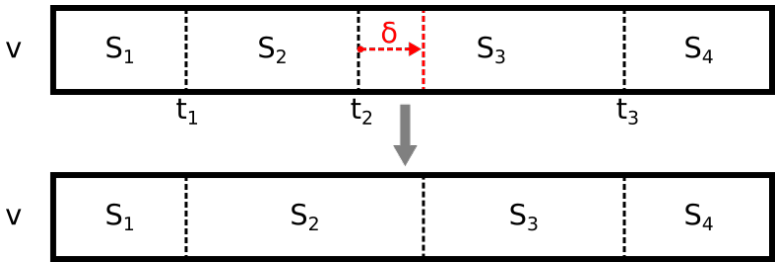

Figure 4.5: **Perturbation move (sec. 4.4).** *In this example, we propose moving the delimiter t_2 between temporal segments s_2 and s_3 by an offset δ (with the constraint that $t_1 < (t_2 + \delta) < t_3$).*

with probability

$$A(\Theta', \Theta) = \min\left(1, \frac{p(\Theta'|D) \cdot q(\Theta'|\Theta)}{p(\Theta|D) \cdot q(\Theta|\Theta')}\right) \tag{4.6}$$

If Θ' is accepted, it becomes the current sample, otherwise we keep Θ. Our proposal distribution uses four different kind of moves, each sampling over a subset of Θ. For each move, we change a single model parameter while keeping all other parameters fixed.

Perturbation move. We define a perturbation as changing the position of one of the current delimiters t by an offset δ (fig. 4.5). We construct Θ' from the current sample with $f(\Theta, t, \delta)$, which replaces t with $t + \delta$. We choose (t, δ) from the space of all possible perturbations (t', δ') conditioned on the current positions of the delimiters in

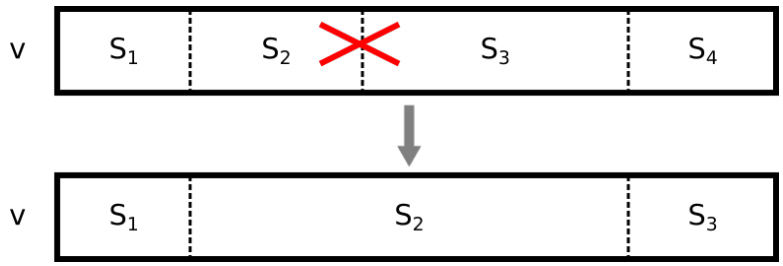

Figure 4.6: **Merge move (sec. 4.4).** *In this example, temporal segments s_2 and s_3 are merged into a single segment.*

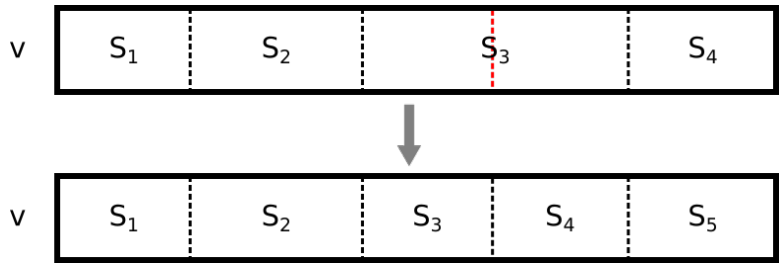

Figure 4.7: **Split move (sec. 4.4).** *In this example, temporal segment s_3 is split in half, creating two new segments.*

Θ. For this, we sample from

$$q_P(t,\delta|\Theta) = \frac{p(f(\Theta,t,\delta)|D)}{\sum_{(t',\delta')} p(f(\Theta,t',\delta')|D)} \tag{4.7}$$

Merge and split move. The merge move proposes merging a pair of subsequent segments into a single one (fig. 4.6). We select which segments to merge from all possible merges given the delimiters in the current sample, using a proposal constructed analogously to (4.7). The complementary split move proposes splitting a segment in half, yielding two segments (fig. 4.7). Both merge and split moves change the number of segments in a video. We note that merge/split moves are quite a standard tool in MCMC methods that require to solve an association problem, for example in the domain of tracking multiple objects (*e.g.* MCMCDA (Oh et al., 2009) or (Brau et al., 2013)).

Correspondence move. This move chooses a segment s_i in a video and proposes to change its matching segment in the other video (*i.e.*, it changes l_i, fig. 4.8). We choose s_i and the new value for l_i from all possible alternatives given the current segmentation $\mathcal{S}$. Again, we use a proposal constructed analogously to (4.7).

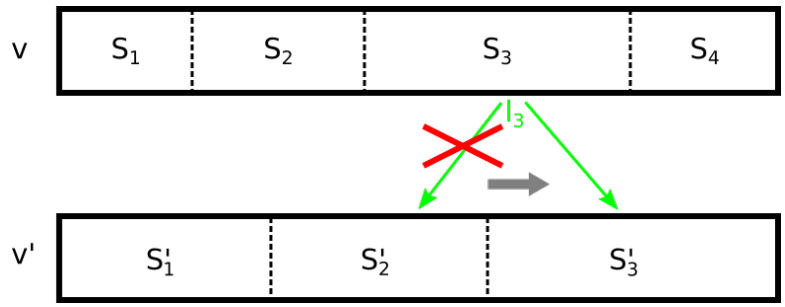

Figure 4.8: **Correspondence move (sec. 4.4)**. *In this example, the correspondence l_3 for temporal segment s_3 (green arrow) changes from s'_2 to s'_3.*

The way we constructed the proposals above increases the acceptance ratio of the moves, which improves mixing. For example, choosing (t, δ) in the perturbation move from a uniform distribution would result in a low acceptance ratio (which significantly improves using q_P). Note that, while our proposals need to compute a large number of posteriors during each move, they can still do it efficiently since most of the terms are shared between these computations, and need to be computed only once.

Initialisation. Starting from a Θ sample with a reasonably good posterior reduces the amount of time wasted in regions of low probability at the beginning of the sampling process (compared to random initialisation). We begin by individually decomposing each video into homogeneous temporal segments, without considering any correspondences. This is achieved by optimising the temporal segmentation likelihood $p(\mathcal{S}|D)$ using just perturbation, merge and split moves. Since there is no correspondence likelihood involved, $p(\mathcal{S})p(\mathcal{S}|D)$ can be optimised independently for each video.

Having the initial temporal segmentation $\mathcal{S}$, we then find the optimal correspondence between these segments. This corresponds to optimising the correspondence likelihood $p(\mathcal{L}|\mathcal{S}, D)$. Since the correspondences are conditionally independent under our model, we can find the exact optimal set of correspondences with a nearest neighbour approach.

4.5 Appearance descriptors

We now discuss the appearance distance a that we use to compute the distance between frames as part of our likelihood (sec. 4.3). We designed it to capture the difference in viewpoint between two frames.

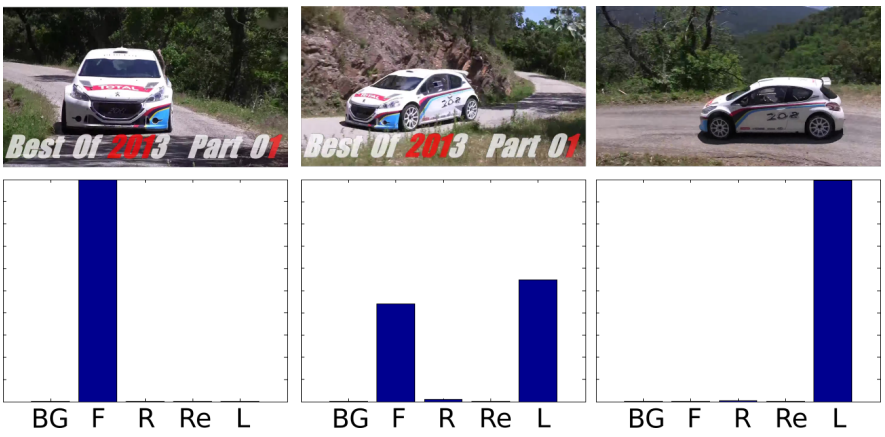

Figure 4.9: *Three example outputs of the softmax layer from cars in* 90*,* 150 *and* 180 *degree viewpoints respectively. The labels correspond to background (BG), front* 90 *degrees (F), right* 0 *degrees (R), rear* 270 *degrees (Re) and left* 180 *degrees (L).*

Appearance distance. Modern appearance classifiers based on Convolutional Neural Networks (CNN) are state-of-the-art for whole image classification (Krizhevsky et al., 2012) and object detection (Girshick et al., 2014). However, they are optimised to differentiate between objects of different classes, and they actually strive for invariance to viewpoint differences. Therefore they are not ideal for our problem. Instead, we train a CNN classifier to distinguish among viewpoints of the car class (which we use in our experiments). We start from a CNN pre-trained for image classification on the ILSVRC 2012 (Krizhevsky et al., 2012; Russakovsky et al., 2015) and finetune it to classify 4 car viewpoints (front, left, right, rear) and the background. As training data we use the PASCAL VOC 2012 (Everingham et al., 2012) dataset which has car images with viewpoint annotations. In order to focus on the appearance of the cars and not on the background, we crop the cars from the image using the provided bounding-box annotations.

After training, we apply the CNN viewpoint classifier on a video frame and use the output of the softmax layer as our frame descriptor (i.e. a 5D vector, summing to 1). The intuition is that if the viewpoint of the input frame matches one of the training viewpoints from PASCAL VOC closely, the softmax output vector will be peaked on one of the 4 viewpoints. Instead, if the viewpoint of the frame lies in-between the training viewpoints, the output probability mass should be spread between two viewpoints (fig. 4.9). We then define the distance a between two frames as the histogram intersection of their frame descriptors.

Figure 4.10: ***A visualisation of the segmentation pipeline*** *We segment the cars in the videos using video foreground segmentation (Papazoglou and Ferrari, 2013) (sec. 4.5). Here we show: object proposals from selective search (Uijlings et al., 2013) (top left), the pixel-wise probability map produced by the car detector (**?**) (top right), the resulting segmentation (bottom left), and the bounding box of the segmentation is used to extract the viewpoint descriptor (bottom right).*

Object localisation in the videos. In order to compute the viewpoint descriptor on a video frame, we first need to localise the car up to a bounding-box. This focuses the descriptor on the appearance of the car and matches the kind of data the CNN was trained on.

To do so, we use the class-specific segmentation method presented in chapter 3, which incorporates a car detector trained on PASCAL VOC 2012 dataset (Everingham et al., 2012). As shown, the class-specific segmentation method produces superior results compared to both the class-agnostic methods and the single object detector (sec. 3.4). For this particular dataset the class-specific segmentation method has an accuracy of **88**.**5**% according to the CorLoc measure (Prest et al., 2012), versus **73**.**2**% for the class-agnostic segmentation method (chapter 2). Interestingly, the class detector (Girshick et al., 2014) alone achieves just 69.8%, confirming once more that using video object segmentation can significantly improve the accuracy over using just a detector (sec. 3.4).

4.6 Experimental evaluation

In this section we first introduce the data used for evaluation (sec 4.6.1). Second, we present the methods we compare against (sec. 4.6.2). Next, we present our evaluation protocol (sec.4.6.3) and finally discuss our results (sec 4.6.4).

4.6.1 Data

We assembled a novel dataset of 22 video sequences ($2-43$ seconds each) depicting cars on racing sequences collected from YouTube. These videos are challenging, showing different cars in different races, with fast motion, fast moving camera and cluttered backgrounds. We automatically split them into different shots using (Kim and Kim, 2009). Each shot is 5-30 seconds long.

Our data contains viewpoint annotations for each frame. For this, we use an annotation protocol that reduces the amount of manual effort as follows. We first define a set of 16 canonical viewpoints, spaced by 22.5 degrees (starting from full frontal). We manually annotate all the frames showing one of them. Then, we automatically annotated the rest of the frames by linearly interpolating the manual annotations. We will soon release all the video sequences and annotations on our website.

We evaluated our model on all pairs v, v' of videos that overlapped by at least 50%, in terms of the percentage of frames in v that show viewpoints that also appear in $v\prime$ (within a margin of 10 degrees). This leads to 251 pairs of videos out of the 484 possible pairs. We plan to release this dataset along with the ground truth annotations.

4.6.2 Alternative methods

We compare our model against three alternatives: a nearest neighbour model, a Markov Random Field (MRF) model promoting temporal smoothness, and Dynamic Time Warping (DTW). Note that as input, all methods use the same appearance distance a (sec. 4.5), same with our model.

Nearest neighbour. The nearest neighbour model matches each frame in v to its closest neighbour in $v\prime$ according to the appearance distance a (sec. 4.5). This simple model has no notion of temporal smoothness or consistency, and it allows us to verify what we can achieve using appearance alone.

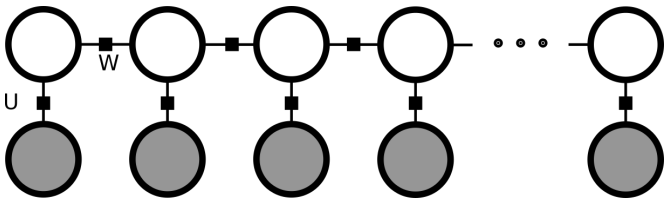

Figure 4.11: The graphical model corresponding to eq. 4.9. Each hidden node (in white) corresponds to a single frame in video v. The observable states (in grey) correspond to the similarity to frames in video $v\prime$.

This approach corresponds to the following model:

$$\mathcal{L}^* = \underset{\mathcal{L}}{\operatorname{argmin}} \sum_f U(l_f) \tag{4.8}$$

where U is a unary cost of matching frame f in v to a frame $f\prime$ in $v\prime$. Note that in this case, l denotes the correspondence between individual frames instead of entire temporal segments.

MRF model. As a second method, we augment the nearest neighbour model by adding temporal smoothness between subsequent frames. For this we use an Markov Random Field (MRF) with pairwise terms that promote that consecutive frames in v are in correspondence with frames in v' that are also close in time (fig. 4.11). More precisely, we solve the following optimisation problem

$$\mathcal{L}^* = \underset{\mathcal{L}}{\operatorname{argmin}} \sum_f U(l_f) + \alpha_W \sum_f W(l_f, l_{f+1}) \tag{4.9}$$

where α_W is a weight term, and U is the same unary term used in the nearest neighbour model. We define the pairwise potential W to be

$$W(l_f, l_{f+1}) = (|l_f - l_{f+1}| - 1)^2 \tag{4.10}$$

Dynamic Time Warping. Dynamic Time Warping (DTW) (Sakoe and Chiba, 1978) is a popular sequence alignment algorithm. Assuming that the two sequences show the same event transition with the only variable being the speed of each, DTW can compute an optimal alignment between them. However, this assumption does not necessarily hold true for realistic video sequences. As input, we used the appearance distance a to compute the distance between individual frames.

Method	NN	MRF	DTW	ours
NN	-	12.7	34.8	9.1
MRF	87.3	-	47.2	15.6
DTW	64.2	52.8	-	23.4
ours	**90.9**	**84.4**	**76.6**	-

Table 4.1: **Comparative user study.** *The table shows the comparative results between the different methods: nearest neighbour (NN), the MRF model, DTW and our model. The value in a cell shows the percentage of videos for which the humans preferred the method of the corresponding row over that of the column. For example, the participants preferred our method over the nearest neighbour approach for* 90.9% *of the videos.*

4.6.3 Evaluation protocol

Comparative user study. We performed a substantial user study to verify that our results are indeed visually more appealing to humans compared to the alternative methods. We performed a "blind taste test" in which users are presented the same sequences aligned by two different methods, and asked which alignment they think is better, *i.e.*, it is consistent in terms of viewpoint and also looks pleasing (smooth frame transitions and less stuttering). In our setup the user is shown an original video and how it was aligned to a second video by two different alignment methods. The original video is displayed in the centre of the screen, the two alignments are on each side, being played simultaneously (we randomly choose on which side we put them). The user has then to decide which one they think is better. Note that we never reveal to the user which method was used to produce the alignments we display. The exact instructions given to the users are given in appendix A.

We use this protocol to compare all of the alignment methods in pairs (*e.g.* our full method vs DTW, DTW vs MRF, etc.). We ensure that each pair of methods is shown to at least 3 different users for each pair of videos and we aggregate the results (table 4.1).

Quantitative criteria. We identify several properties that correspond to what humans perceive as attractive alignments. First, frames in correspondence should be displaying the same viewpoint. Second, the viewpoint transitions should be temporally smooth. Third, long sequences of correct correspondences are preferable. Based on this observation, we propose two measures to evaluate an alignment quantitatively:

Percentage of correct correspondences: This measures the percentage of frames that are in correct correspondence, *i.e.* difference in viewpoint is 22.5 degrees or less

Measurement	Correct corresp. %	Longest correct seq.
NN	29.8	9.4
2nd order	46.0	27.5
DTW	40.8	26.4
ours	**47.8**	**30.3**

Table 4.2: **Quantitative results.** *Comparison of the different methods: nearest neighbour (NN), the MRF model, DTW and our model. The first column is the percentage of correct correspondence, while the second column is the longest correct sequence (sec. 4.6.3).*

(which is equal to the spacing of our manually annotated canonical viewpoints), and the difference in the viewpoint derivative is 5 degrees/frame or less. Intuitively, the viewpoint difference ensures that aligned frames show the same viewpoint, while the derivative difference ensures that the viewpoint transition is smooth.

Longest correct sequence: This measures the length of the longest sequence of correct correspondences in each video, normalised by the length of the video. Intuitively, alignments that are correct for large periods of time are visually preferable to alignments with alternating correct and erroneous correspondences.

We analyse how well these two evaluation criteria capture what humans perceive as a good temporal alignment, by verifying how accurately they can predict the results of the user study. We do this as follows. Given two methods, we predict that the user will choose the alignment found by the method that scores higher according to the evaluation criteria. We then report the *prediction accuracy* of each criterion, *i.e.* the number of times the prediction made using that criterion is correct, averaged over all possible pairs of methods and videos (table. 4.3).

4.6.4 Results and discussion

Comparative user study. Table 4.1 shows the results of the user study. The value in a cell shows the percentage of videos for which the humans preferred the method of the corresponding row over that of the column. Our method substantially outperforms all three alternative methods (last row).

The nearest neighbour model produces very jittery alignments, as it does not enforce any temporal smoothness. As a consequence, the users do not find the output visually pleasing. Thanks to pairwise temporal smoothness, the MRF model partly alleviates this problem. However, the smoothness is promoted only at a local level (be-

	Human agreement
Correct correspondence %	70.7
Longest correct sequence	77.7
Human judgement	92.08

Table 4.3: **Evaluation of criteria.** Each value corresponds to the accuracy of each criterion when trying to predict the results of our user study (sec. 4.6.3). The last row is the agreement between the human evaluators and acts as an upper limit to the predictive ability of a criterion.

tween consecutive frames). Hence, the MRF is unable to capture smooth transitions of viewpoints on a larger time scale. Instead, our model enforces smoothness at the level of temporal segments, leading to large, piece-wise smooth alignments.

Interestingly, the users clearly prefer DTW over the nearest neighbour model, but results are comparable with respect to the MRF model. As mentioned before, DTW makes the strong assumption that both videos show the exact same sequence of events, possibly occurring at varying speeds. When this assumption holds, DTW can produce an optimal temporal alignment, and the users prefer it over MRF. However, in the scenario where this assumption does not hold, the users consistently prefer MRF. They however clearly prefer our method over both DTW and MRF, as our model can handle both scenarios thanks to the temporal segmentation.

Quantitative criteria. Table 4.2 shows the performance of each method according to our two quantitative criteria (sec. 4.6.3). Our method outperforms all of the alternatives for both criteria In contrast to the human study results, DTW performs significantly worse than the MRF model with respect to percentage of correct correspondence. This indicates that humans tend to prefer smoother alignments, even if the aligned frames exhibit a larger difference in viewpoint. While the quantitative performance difference between the MRF model and ours is relatively small, the users still prefer our method.

Quantitative criteria vs user study. Table 4.3 shows an analysis of how well our evaluation criteria can predict what humans perceive as visually pleasing alignments. Each value corresponds to the prediction accuracy of each criterion (sec. 4.6.3). As can be seen from the results, both criteria show a strong correlation to what the users prefer, in particular the longest correct sequence. It is worth noting that absolute agreement is not achievable, as there is disagreement even among human evaluators. This is especially true in cases where both methods produce bad alignments, which makes it

Figure 4.12: ***Qualitative results.*** *Each pair of rows (a-d) shows an original video (top) and a second video aligned to it by our method (bottom). Notice that in (a) the video has to be played backwards after the middle of the sequence for a correct alignment. This would not be possible with DTW.*

hard for the evaluators to pick the better one. As shown in table 4.3 we computed the agreement between human evaluators which acts as an upper bound to the predictive ability of any criterion.

We present some qualitative results of our method on various pairs of videos (fig. 4.12). Videos showing some example results are included in the supplementary material.

Chapter 5

Discovering object aspects from video

5.1 Introduction

Traditionally, visual aspects have been defined as distinct viewpoints of rigid 3-D objects (Koenderink and van Doorn, 1979; Plantinga and Dyer, 1986; Bowyer et al., 1988; Cyr and Kimia, 2001). However, viewpoint alone cannot capture the appearance variations of complex, articulated objects in natural images.

For example, tigers seen from a similar viewpoint can look very different due to articulated pose (*e.g.* a tiger lying and a tiger standing, fig. 5.1). We use a broader notion of aspect that considers four factors of variation: viewpoint, articulated pose, occlusions and cropping by the image border. We explore the problem of automatically discovering such aspects from natural images of an object class. This task requires finding different object instances showing the same aspect (*e.g.* tigers running to the right, face close-ups, fig. 5.1).

While some recent methods discover aspects from *still images* (Felzenszwalb et al., 2010; Gu and Ren, 2010; Divvala et al., 2012; Drayer and Brox, 2014; Dong et al., 2013; Aghazadeh et al., 2012), they all require manual annotations of the object's location (*e.g.* bounding-boxes). Location annotations allow focussing on the appearance of the object rather than the background, but they are expensive and time-consuming to create. In this work instead we discover aspects from *video*, where we can segment the foreground objects from the background automatically, by exploiting motion (chapter 2). Hence, it is possible to discover aspects under weak supervision, *i.e.* only the class label of each video shot is required. As an additional advantage, we can easily obtain video data for a large number of classes from several sources (*e.g.* DVDs, YouTube).

We present an extensive exploration of weakly-supervised aspect discovery in video, which we pose as an image clustering problem (sec. 5.5). We measure the quality of the discovered aspects in terms of the compactness and diversity of the clustering (sec. 5.6.1). We experiment with several modern appearance descriptors (SIFT (Lowe, 2001), shape contexts (Belongie and Malik, 2002), CNN features (Jia, 2013)), and various levels of spatial support (e.g. whole image, foreground segmentation). This enables to carefully evaluate the benefits of automatically segmenting objects (sec. 5.6).

Our exploration relies on a new protocol for evaluating aspect discovery directly. In contrast, previous works evaluate aspect discovery indirectly, typically by measuring its impact on object detection performance (Felzenszwalb et al., 2010; Gu and Ren, 2010; Divvala et al., 2012; Drayer and Brox, 2014). For this, we collected a large dataset sourced from videos of two different classes, car and tiger (for a total of 2664 video shots, sec. 5.4). The choice of the car and tiger classes allows us to explore two very different scenarios. Cars are rigid objects, and the major factors of aspect variations are different viewpoint, occlusions and croppings. Tigers display a broader range of different poses due to their complex articulation (Fig. 5.1). As an additional difference, cars exhibit higher intra-class variability in color and shape than tigers (*e.g.* different makes).

We annotated a few frames per shot with ground-truth aspect labels using an efficient labelling scheme (totalling over $10,000$ frames, sec. 5.3). This scheme captures the four factors of aspect variation by labelling simple, discrete properties of the object's physical parts. For example, we can distinguish between the top two aspects in fig. 5.1 by considering that the hind legs are not visible in the second. We plan to release this dataset and the aspect labels.

Our experimental exploration demonstrates the great potential of using video for weakly supervised discovery (sec. 5.6). In particular, the accuracy of the discovered aspects improves significantly if we use motion segmentation to get an estimate of the object location. After evaluating aspect discovery directly, we also show that it is useful for other applications. First, we use the aspects discovered by our system to enable a new kind of image retrieval based on aspects (sec. 5.7.1). Second, we exploit the temporal nature of video to learn models of aspect transitions (*e.g.* from lying to standing, sec. 5.7.2).

The rest of the chapter is organized as follows. We start by discussing the two main components of our evaluation protocol: the labelling scheme (sec. 5.3) and the dataset (sec. 5.4). We then present several strategies for aspect discovery (from both videos and

Figure 5.1: **Aspects discovered by our method (one per row).** *Despite showing tigers from the same viewpoint, the top two aspects look very different due to articulated pose and cropping. Our notion of aspect considers all these factors (sec. 5.3).*

still images, sec. 5.5) and present the results of our extensive exploration (sec. 5.6). We conclude by introducing two applications that benefit from aspect discovery (sec. 5.7).

This work has been published in (Papazoglou et al., 2016).

5.2 Related Work

Early work on aspects Early work considered simple objects for which all possible aspects could be exhaustively enumerated (Koenderink and van Doorn, 1979; Plantinga and Dyer, 1986; Bowyer et al., 1988). More recently, (Cyr and Kimia, 2001) tried to learn a manageable collection of representative views of an object instance. All these methods are limited to synthetic views of a single object instance.

Aspect Discovery Several methods (Felzenszwalb et al., 2010; Gu and Ren, 2010; Divvala et al., 2012; Drayer and Brox, 2014; Dong et al., 2013; Aghazadeh et al., 2012; Azizpour and Laptev, 2012; Bourdev et al., 2010; Gu et al., 2012) discover aspects implicitly, in order to train specialised classifiers for each of them (components of a mixture model). Some of these works (Felzenszwalb et al., 2010; Gu and Ren,

2010; Divvala et al., 2012; Drayer and Brox, 2014) cluster HOG descriptors extracted from bounding-boxes in the training images (manually annotated). Others (Dong et al., 2013; Aghazadeh et al., 2012) use exemplar SVMs (Malisiewicz, 2011) as a similarity measure between bounding-boxes to drive the clustering. A few methods require additional time-consuming annotations, such as the location of object parts (Azizpour and Laptev, 2012) or keypoints (Bourdev et al., 2010; Gu et al., 2012). None of the methods above is weakly supervised. Moreover, while aspect discovery is a crucial intermediate step in their pipeline, it is evaluated only indirectly by measuring the performance improvement of the overall system.

Aspects in multi-view models The works above use the discovered components in isolation. In contrast, other methods take the relationships between different aspects into account to build multi-view models (Savarese and Fei-Fei, 2008; Thomas et al., 2006; Mei et al., 2011; Liebelt et al., 2008; Liebelt and Schmid, 2010). They either require expensive bounding-box and viewpoint annotations for each training image (Savarese and Fei-Fei, 2008; Thomas et al., 2006; Mei et al., 2011) or very detailed 3-D CAD models (Liebelt et al., 2008; Liebelt and Schmid, 2010). Only the work of (Su et al., 2009) uses video for this task. Their method is trained on a single short cellphone video per class, taken by walking around the object. While this procedure captures viewpoints well, it might fail to record other factors of variation, such as articulated pose. Moreover, it is not easily applicable for certain classes, such as wild animals. In practice, (Su et al., 2009) only considers common rigid objects *i.e.* cars, motorbikes, wheelchairs, etc.

Modelling pose variations with parts In the context of object detection and segmentation, some works (Bourdev et al., 2010; Bourdev and Malik, 2009; Brox et al., 2011) model variations in pose and articulation using poselets, *i.e.* parts that are tightly clustered in both appearance and configuration space (e.g crossed hands, frontal face). This is somewhat related to our definition of aspects in terms of part properties (sec. 5.3). However, learning poselets requires manual annotation of keypoints (Bourdev et al., 2010; Bourdev and Malik, 2009) and 3-D joint configurations (Bourdev et al., 2010), so they are not suitable for weakly supervised aspect discovery.

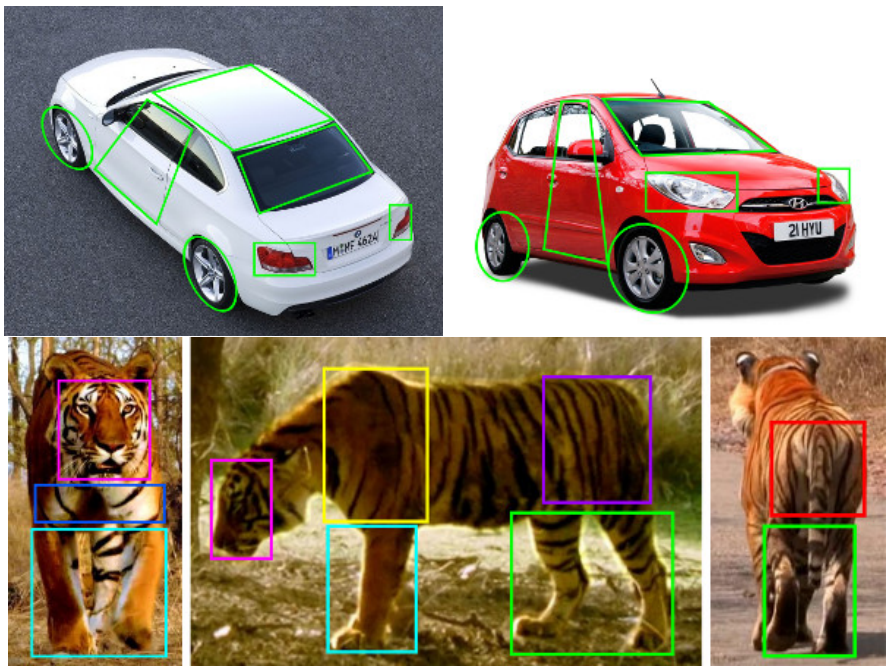

Figure 5.2: **Part visibility labels.** ***(Top)*** *We annotate 13 physical parts of cars with visibility tags (sec. 5.3).* ***(Bottom)*** *We annotate 9 physical body parts of tigers with visibility tags. Note that our annotation is weak, we do not mark the parts with bounding-boxes.*

5.3 Aspect labels

Our labels accurately capture the four factors of aspect variation (viewpoint, articulated pose, occlusions, cropping), by considering simple properties of the object's physical parts (*e.g.* head, legs *etc.*, fig. 5.2, 5.3). We uniquely identify the viewpoint, occlusions and cropping by considering which parts of the objects are visible in the image (e.g. when a tiger is seen from the back, the face is not visible, fig. 5.2). We capture pose variations using additional configuration labels for the articulated parts (*e.g.* standing, lying for legs).

This scheme provides a compact yet fine-grained description of the object's aspect. As an additional advantage, it is easy to annotate accurately and unambiguously. Moreover, it naturally allows us to define a distance between aspects, which we will use for evaluation.

Part visibility labels For cars we use 13 parts: windscreen, wheels, lights, frontal doors and roof (fig. 5.2 top). For tigers we use 9 parts: face, sternum, left and right shoulders, left and right thighs, front and hind legs, and buttocks (fig. 5.2 bottom). We

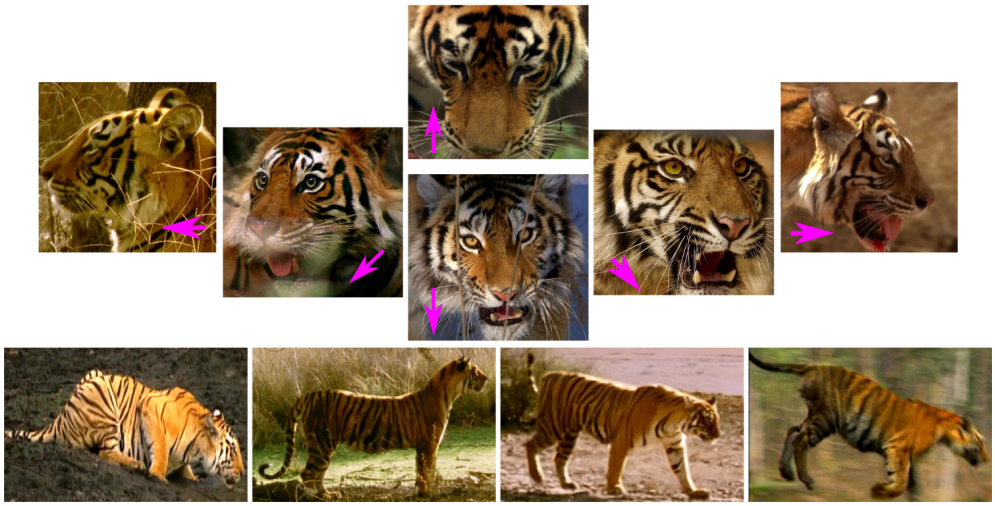

Figure 5.3: **Part configuration labels (for tigers only).** ***(Top)*** *We annotate 6 different face orientations (sec. 5.3).* ***(Bottom)*** *We annotate 4 different leg configurations.*

annotate a part as visible if more than 50% of the area of that part is visible.

Part configuration labels For tigers, we choose the orientation of the face from six possible orientations (when visible, fig. 5.3 top). This allows to distinguish across different face close-ups, which are very frequent in animal videos. We also choose the leg configuration from: lying, standing, walking and running (fig. 5.3). This property is indicative of both pose and appearance (due to motion blur). It is significantly easier and less time-consuming for humans to annotate than, say, specifying the angles of the joints of the leg.

Distance between aspects We now define a distance to measure the similarity between two aspects. For instance, walking to the right should be closer to running to the right than a face close-up. Standing facing right should be closer to laying facing right than to laying facing towards the camera. Our distance captures such transitions in aspect space smoothly by using the part labels. We argue that this is much more expressive than considering aspects as mutually exclusive categories, which would require complex hand-defined rules to determine the distance between aspect categories.

Let A_i and A_j be two aspects. We define:

$$D(A_i,A_j) = 1 - \sum_p d_p(A_i,A_j) \Big/ \left|V(A_i) \cup V(A_j)\right| \tag{5.1}$$

where d_p is the distance with respect to part p, and $V(A)$ the set of visible parts in A;

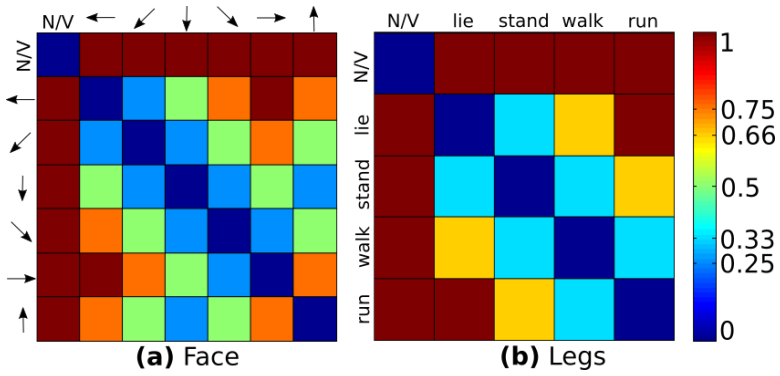

Figure 5.4: **Distance matrices for the "face" and "legs" parts.** ***(a)*** *Distance matrix for the "face" part (d_p, sec. 5.3). The entries show the distance between the different face orientations, denoted by the arrows (fig. 5.3 top). N/V denotes that the part is not visible.* ***(b)*** *Distance matrix for the "legs" part (d_p, sec. 5.3). The entries show the distance between the different leg configurations (fig. 5.3 bottom).*

$d_p(A_i, A_j) = 1$ if p is visible in both aspects, 0 otherwise. For face and legs, d_p further depends smoothly on the difference in orientation/action (fig. 5.4).

5.4 Dataset

We assembled a dataset containing several hundreds video shots for two different classes (car and tiger). We annotated frames in each shot with the *aspect labels* (sec. 5.3), which allows direct evaluation of aspect discovery (sec. 5.5). Finally, we exploit the nature of video to provide automatic object localisation for each frame using foreground segmentation through motion.

We collected the shots from 188 car ads (~1-2 minutes each) and 14 nature documentaries about tigers (~40 minutes), amounting to roughly 14 hours of video. We automatically partitioned these raw videos into shorter shots (Kim and Kim, 2009), and kept only those showing at least one instance of the class. This produced 806 shots for the car and 1880 for the tiger class, typically $1-100$ seconds in length.

We annotated aspect labels as follows. First, we randomly chose five frames per shot, and annotated each of them with the number of objects shown. We then gave aspect labels only to frames showing exactly one object (to avoid ambiguities). This produces a total of 6610 frames with aspect label for tigers, and 3485 for cars.

Last, we used the class-agnostic segmentation method (chapter 3) to automatically segment the foreground in each shot. For the frames with aspect labels we also marked

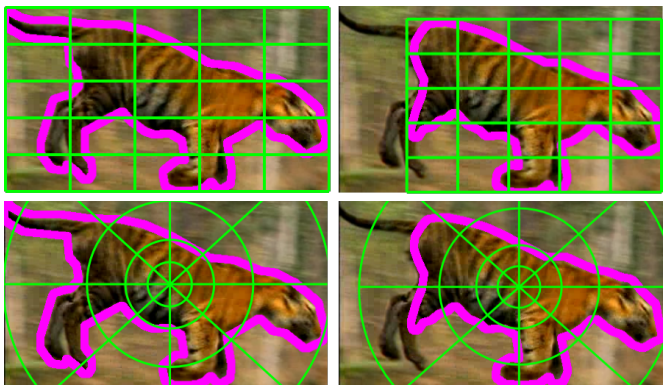

Figure 5.5: **Spatial binning for BoVW descriptors. (Top)** *A rectangular grid fit over the segmentation (Papazoglou and Ferrari, 2013). Even small segmentation errors (right) lead to a very different configuration of the spatial bins.* **(Bottom)** *A log-polar grid placed on the centre of mass of the segmentation. Both the centre of mass and the radius of the log-polar grid are robust to small segmentation errors (sec. 5.5.1).*

whether the segmentation is *accurate*.

Statistics For the aspect labels, we observed 643 unique combinations for the tigers, and 293 for cars. Some are more frequent, for example there are 221 frontal face close-ups. Last, the segmentation is accurate in 55% of the frames, which is in line with the results reported for the YouTube-Objects dataset (sec. 2.4.3).

5.5 Automatic aspect discovery from video

We treat aspect discovery as a frame clustering problem. We explore two families of descriptors: bag-of-visual-words (BoVW, sec. 5.5.1) and Convolutional Neural Networks (CNN, sec 5.5.2). We consider various spatial supports over which to compute descriptors, including the whole frame or the foreground segmentation produced automatically by (Papazoglou and Ferrari, 2013).

5.5.1 Bag-of-Visual-words descriptors

The Bag-of-Visual-Words (BoVW) approach models an image as an orderless collection of *visual words* (*i.e.* quantized local features). The BoVW descriptor is a histogram recording the frequencies of the visual words over a *spatial support* of interest (*e.g.* an entire image or an image region). While this disregards information about

the spatial layout of the image, adding geometric information using *spatial binnings* (fig. 5.5) can help image classification (Lazebnik et al., 2006) and object detection (Uijlings et al., 2013) performance.

We consider various combinations of visual words (SIFT (Lowe, 2001) and shape-contexts (Belongie et al., 2002)), spatial supports (*e.g.* foreground segmentation (Papazoglou and Ferrari, 2013)), and spatial binnings (*e.g.* spatial pyramids (Lazebnik et al., 2006)). Each combination produces a different BoVW descriptor.

Visual words First, we consider dense *SIFT* (Lowe, 2001) computed on 4x4 pixel patches at every pixel. Second, we use (Belongie et al., 2002) to extract shape-context features from the contour of the segmentation. We convert these features into visual words using a vocabulary of 1000 visual words for *SIFT* and 100 for shape-contexts.

Spatial support We consider three types of spatial supports to determine the extent to which each feature contributes to the BoVW: *whole frames*, *segmentation* (Papazoglou and Ferrari, 2013), and *motion saliency* (Papazoglou and Ferrari, 2013). We use a general, uniform treatment for all supports, by assigning a weight $w_i \in [0,1]$ to each pixel i in the frame. The feature at i contributes by w_i to the BoVW.

For whole frames, we give equal weight to all pixels (*i.e.* $w_i = 1 \forall i$). For the segmentation we set $w_i = 1$ if i is part of the foreground, otherwise $w_i = 0$. Motion saliency uses motion to compute the probability p_i that pixel i is part of foreground (we simply set $w_i = p_i$). This can be seen as a soft version of the segmentation. Typically, it produces a roughly correct localisation even when the segmentation is very inaccurate (fig. 5.6).

Since shape-contexts are defined on object contours, we only use them with the segmentation (we try all supports for SIFT). Last, note how segmentation and motion saliency enable to measure appearance purely on the object, excluding the background. They are made possible by exploiting the temporal nature of video.

Spatial binning The basic idea of spatial binning is to partition the spatial support into a fixed set of spatial bins, and compute a separate histogram for each. Here, we consider two different variants.

First, we use 3-level spatial pyramids over a *rectangular grid* (Lazebnik et al., 2006). Second, we propose a *log-polar radial* binning inspired by (Belongie et al., 2002). The log-polar bins are placed on the centre of mass of a given spatial support

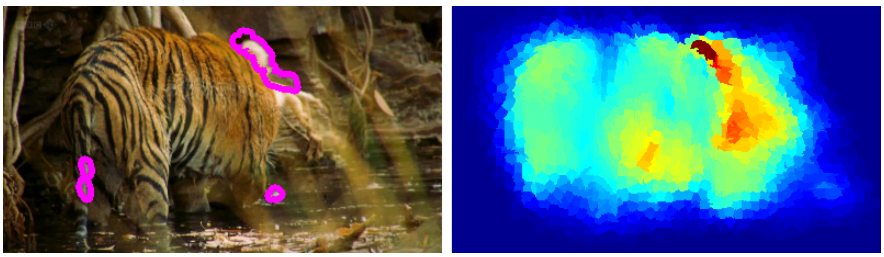

Figure 5.6: **Spatial support.** *Motion saliency (**right**) estimates the probability of being part of the foreground at each pixel (sec. 5.5.1). It often provides a good rough localisation even when the segmentation fails (**left**), where pixels are instead hard assigned to foreground/background.*

(fig. 5.5). We use 8 angular bins and 6 radial bins. To achieve scale invariance, the step of the radial bins is proportional to the scale of the spatial support, *i.e.* the average distance between each pixel i and the centre of mass weighted by w_i. This scheme is more robust to small errors in the segmentations than a rectangular grid (fig. 5.5). Last, we consider orderless BoVWs (*'no binning'*) as a baseline.

5.5.2 CNN descriptors

CNN descriptors achieve state-of-the-art performance on various tasks (*e.g.* classification (Krizhevsky et al., 2012), detection (Girshick et al., 2014)). Like BoVWs, CNN descriptors can be computed on different spatial supports. Note that the concept of binning does not apply here.

Spatial support First, we extract 4096-dimensional CNN descriptors from the whole frame using CAFFE (Jia, 2013). This CNN model was trained for whole image classification on Imagenet (Russakovsky et al., 2015). Second, we extract 4096-dimensional CNN descriptors from the bounding-box of the segmentation. Here, we use a model fine-tuned for object localisation on class-agnostic object proposals (Girshick et al., 2014). We found this to be more suitable for the segmentation support. We do not consider motion saliency, since incorporating individual pixel weights into the CNN framework is not straightforward.

5.5.3 Clustering

We cluster frame descriptors using k-medoids, which is suitable for any distance function. We compute distances between BoVW descriptors using histogram intersection.

For CNN descriptors we use Euclidean distance. For efficiency, we precompute the distance matrix between all frames before clustering. We cluster 1000 times and keep the clustering with the lowest energy to reduce the effects of random initialisation.

5.6 Evaluation of aspect discovery

5.6.1 Protocol

For evaluation, we use two different criteria: clustering energy and diversity. The combination of these two carefully designed measures provides a complete picture of the quality of the clustering.

Clustering energy This measures the compactness of the clusters, *i.e.* it penalises assigning dissimilar aspects to the same cluster. Let A_k be the medoid of cluster k, *i.e.* the aspect in k minimising the sum of distances to all other aspects in k. We define the energy as: $\frac{1}{N}\sum_k \sum_{j\in k} D(A_k, A_j)$, where N is the total number of points being clustered. This is a generalisation of the standard purity evaluation measure (Manning et al., 2008) for a continuous label space, *i.e.* using a smooth D penalises putting items with different labels in the same cluster proportionally to their distance.

Clustering diversity In the video domain, energy can be trivially minimised by clustering together all frames in a shot, which on average contains only 1-2 aspects of the same object instance. Instead, applications using these aspect clusters need to see different object instances of the same aspect (*e.g.* learning a multi-view class model, or retrieving different instances of a query aspect, sec. 5.7.1).

Hence, we also measure the diversity of a cluster, i.e. the average number of different shots per cluster. It rewards clustering together occurrences of the same aspect from different shots (hence different object instances).

5.6.2 Results

We present here an extensive exploration of the various descriptors for aspect discovery (sec. 5.5) on our dataset (sec. 5.4). We evaluate each descriptor separately by computing clustering energy and diversity. Since the true number of aspect clusters is not known a priori we experiment with different numbers of clusters: 50, 100, 200, 400,

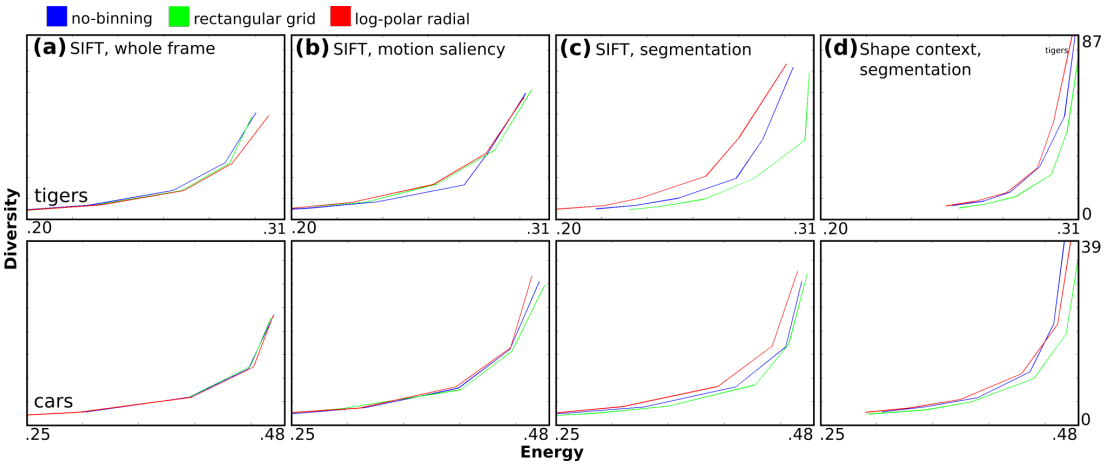

Figure 5.7: *Comparison of different spatial binnings for bags-of-visual-words descriptors (sec. 5.6.2). The first and second row correspond to tigers and cars, respectively.*

600 and 800. Last, we explore learning a better distance for clustering by combining them.

Spatial binning We first evaluate spatial binnings for SIFT on the whole frame (fig. 5.7a). Interestingly, both rectangular grid and log-polar radial are comparable to no binning, which is in contrast to the findings of (Lazebnik et al., 2006) for image classification. This happens because most bins end up covering the background regardless of the choice of spatial binning, when applied to whole frames. On motion saliency (fig. 5.7b), log-polar radial and rectangular grid perform similarly, both being slightly better than no binning. On segmentation, log-polar performs significantly better than rectangular grid for both SIFT (fig. 5.7c) and shape contexts (fig. 5.7d). No binning performs better than rectangular grids, showing that naive rectangular grids are not robust to small errors in the automatic segmentations (fig. 5.5). In all the following experiments we use log-polar binning, as it always performs equally or better than the alternatives.

Spatial support Here we evaluate the different spatial supports. For the SIFT descriptors (fig. 5.8a), both segmentation and motion saliency outperform whole frame, with segmentation offering the best performance. This is because it allows to focus on the appearance of the foreground object. Instead, whole frame is confused by the background, which has little correlation to the object's aspect. When clustering only the frames with accurate segmentation (sec. 5.4), the segmentation spatial support out-

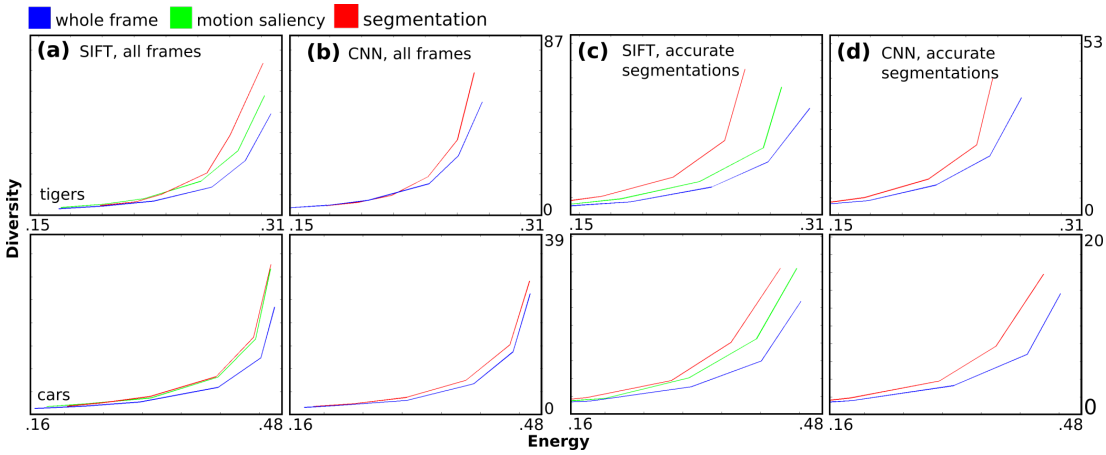

Figure 5.8: *Comparison of different spatial supports (sec. 5.6.2). All SIFT plots (a,c) use log-polar binning. The first and second row correspond to tigers and cars, respectively.*

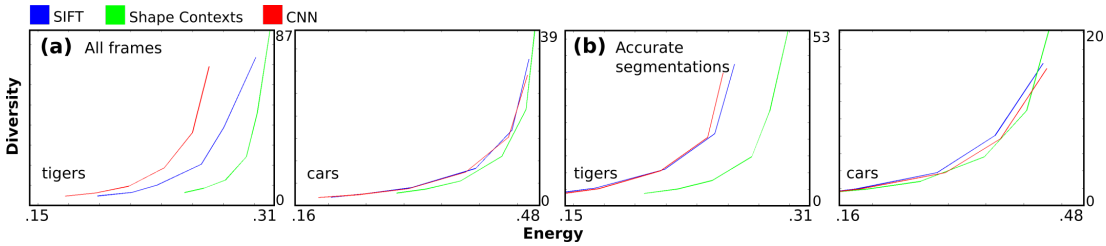

Figure 5.9: *When using segmentation as spatial support, CNN is better or comparable to the other descriptors (**a**). If we evaluate only on frames where the segmentation is accurate (**b**, sec. 5.4), the gap between SIFT and CNN is significantly reduced, especially on tigers (sec. 5.6).*

performs the others by an even larger margin (fig. 5.8c).

Experiments on CNNs reveal the same trend, i.e. segmentation outperforms the whole frame (fig. 5.8b), and the gap between them increases when using only accurate segmentations (fig. 5.8d).

These experiments demonstrate that video offers an advantage over still images as it enables automatic object localisation. Using segmentation improves on the other supports even if it is accurate only half of the time (sec. 5.4). When we focus on frames with accurate segmentations only, the gap increases substantially. This indicates that further advances in video segmentation can lead to even better aspect discovery. Fig. 5.11 and 5.12 show some aspect clusters found using CNN on segmentation. Aspect clusters found using the SIFT BoVW on segmentation are included in the supplementary material.

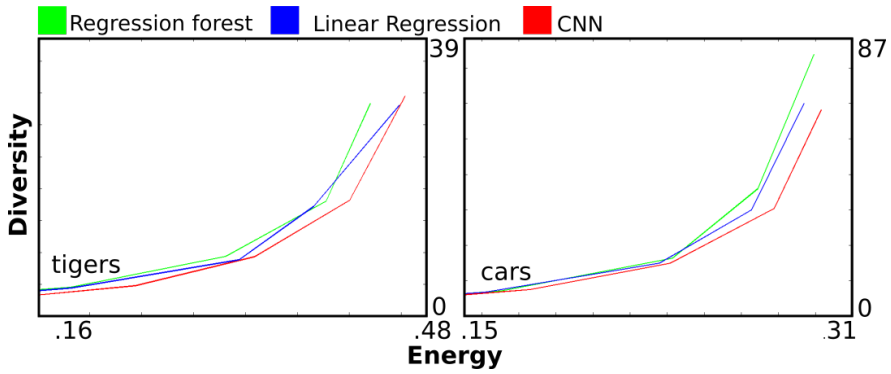

Figure 5.10: ***Distance learning.*** *We learn a distance function that combines all the individual descriptors tested for clustering (sec. 5.6), using two alternative regression methods: linear (pink) and regression forest (cyan). Both outperform CNN on segmentation (red), which is the descriptor that individually performs best (sec. 5.6).*

Descriptors Here we compare the different descriptors (SIFT BoVW, shape-contexts BoVW, CNN, fig. 5.9). For each, we use the best combination of spatial support/binning based on the experiments above.

Shape-contexts is generally inferior to the others, especially on tigers (fig. 5.9b), possibly because the automatic segmentations often miss the fine details of the contours (*e.g.* paws, tail).

CNN outperforms SIFT BoVW significantly on tigers, while they are comparable on cars. When clustering only frames with accurate segmentations, SIFT performs better than CNNs on cars, and is comparable on tigers. This goes against the general trend of CNN outperforming SIFT for various computer vision tasks (Razavian et al., 2014; Donahue et al., 2013; Krizhevsky et al., 2012; Girshick et al., 2014). This might be because CNN do not take full advantage of the detailed pixel-wise support that the segmentation provides, as they are extracted from its bounding-box. Unfortunately, extracting CNNs from a pixel-wise support is still an open problem. Given the ongoing advancements in automatic video segmentation (Lee et al., 2011; Papazoglou and Ferrari, 2013; Faktor and Irani, 2014), this is a promising area to explore.

Distance learning Here, we explore combining all the descriptors mentioned above in order to improve the clustering. Intuitively, we want to drive the clustering with a distance that is as close as possible to the true distance between aspects (5.1). We pose this as a regression problem: we use the distances computed with respect to individual descriptors as predictors, and the distance (5.1) between ground-truth aspect labels as

Figure 5.11: *Example aspect clusters discovered for the tiger class. Each row corresponds to a different cluster. Here, we used CNN on segmentation as descriptors (sec. 5.5.2).*

target.

We begin by splitting the dataset into two halves. We first train a regressor to predict the distance between ground-truth labels (5.1) from the distances of the individual descriptors in one half. Then we use this regressor to predict distances between frames in the other half, and use them for clustering.

We experiment with two alternative regression models, linear regression (Bishop, 2006) and regression forests (Criminisi et al., 2011). Both regressors bring a moderate improvement to using individual descriptors (fig. 5.10). While regression forests provide a better approximation of the ground-truth distance, both methods perform equally well for clustering. Note, however, that this experiment requires aspect labels for the training subset, whereas all the experiments before are unsupervised.

Summary of results The log-polar binning scheme performs best under all circumstances. Segmentation is the best performing spatial support, and in general CNN performs better than SIFT. However, when we focus on videos with accurate segmentation only, the gap between CNN and SIFT disappears. We posit that this happens because CNNs operate on bounding-boxes and cannot fully exploit the pixel-level support provided by the segmentation. Experimenting with accurate segmentation only

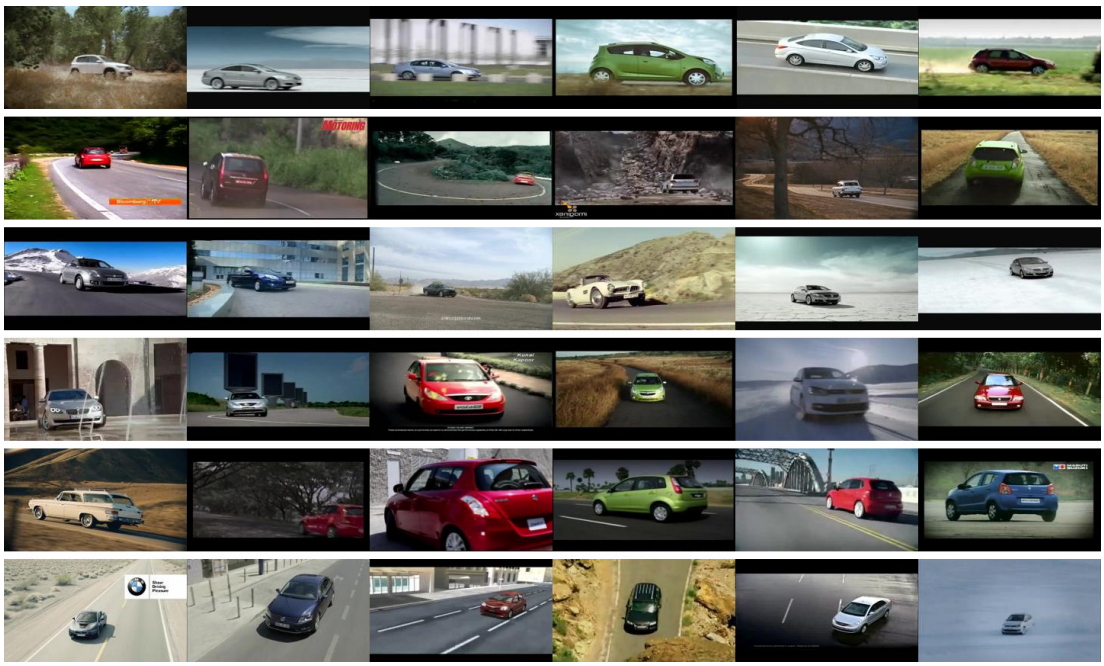

Figure 5.12: *Example aspect clusters discovered for the car class. Each row corresponds to a different cluster. Here, we used CNN on segmentation as descriptors (sec. 5.5.2).*

also indicates that advances in video segmentation will lead to better aspect discovery.

5.7 Applications

We now introduce two possible applications of our aspect discovery system. We discuss an image retrieval system for aspects (sec. 5.7.1), and how to learn transitions in aspect space (sec. 5.7.2).

5.7.1 Aspect image retrieval

We now discuss an image retrieval application that exploits the aspect clusters discovered by our method. Specifically, we build an "aspect retrieval" system, where a user enters a textual query specifying an aspect with a natural semantic label (e.g. frontal tiger, face close-up), and the system automatically retrieves suitable images (fig. 5.13b-d) from a large unlabelled database $\mathcal{D}$ (fig. 5.13a).

To achieve this, the retrieval system needs to learn about the appearance of each semantic label. The traditional way to do it would require labelling a large number of training images per label. Instead, we use as training data a set $\mathcal{V}$ of videos of the class

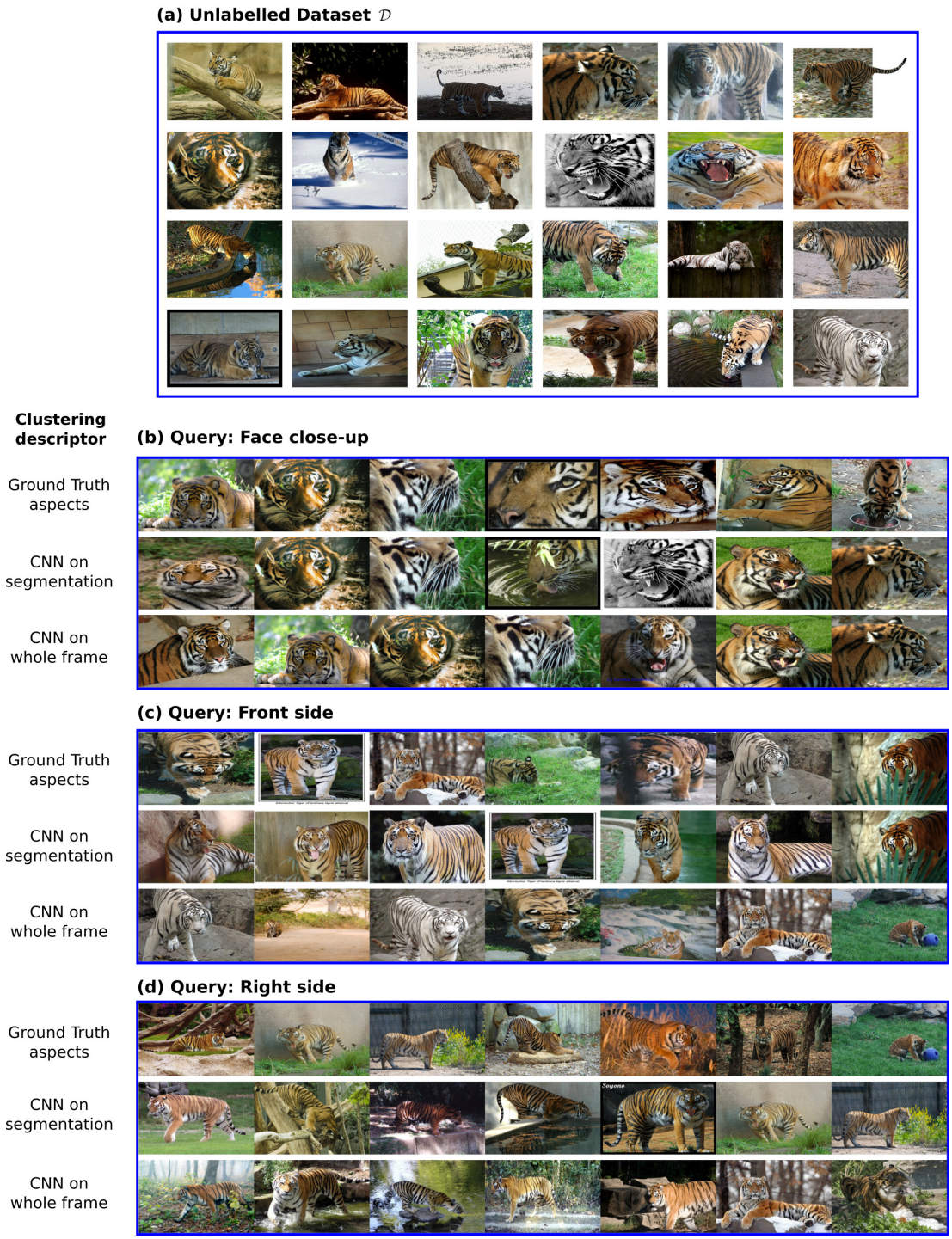

Figure 5.13: *Our aspect retrieval system (sec. 5.7.1) supports searching a database $\mathcal{D}$ of unlabelled images using an aspect semantic label as query (e.g."Right side" or "Face close-up"). Here, we show a subset of $\mathcal{D}$ (a), and some examples of the images retrieved by our system (b-d). We illustrate the 7 highest scoring images for: face close-up (b), front size (c) and right side (d). Each panel (b-d) shows the output of the retrieval system for three different strategies for aspect discovery: using ground-truth aspects, CNN on segmentation, and CNN on whole frame.*

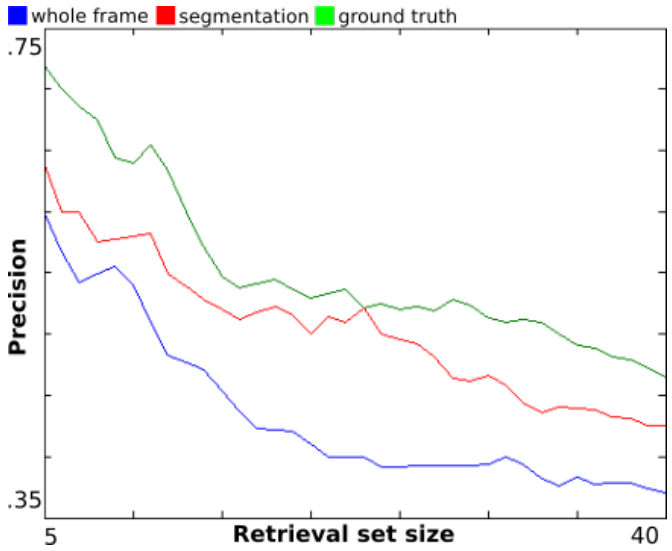

Figure 5.14: ***Results on image retrieval*** *(sec. 5.7.1). The horizontal axis indicates the size of the retrieval set returned to the user. The vertical axis is precision. The curves differ only in the distance used for finding aspect clusters in the video database. The areas under each curve are* 18.85*,* 21.74 *and* 23.9 *for whole frame, segmentation and ground-truth respectively.*

with no semantic labels. First, we let our system discover clusters of aspects in $\mathcal{V}$. The annotator then assigns *one* semantic label to each cluster, which significantly reduces the annotation effort (33$\times$ in the experiments below).

Protocol For this experiment, we define five semantic aspect labels: face close-up, left side, right side, front side and back side. For training, we use the 6610 frames of the tiger class (sec. 5.4) as $\mathcal{V}$. Instead of manually labelling each individual frame, we cluster them automatically (sec. 5.5.3) using CNN on segmentation as descriptor (sec. 5.5.2). We set the number of clusters to 200. We then label each cluster with the most frequent semantic label in it, choosing from the five options above (the label gets assigned to each image in the cluster). This effectively reduces the number of items to manually annotate from 6610 to 200, reducing the human effort by a factor of 33.

For testing, we use a database $\mathcal{D}$ consisting of 200 images of tigers sourced from ImageNet (Russakovsky et al., 2015) (fig. 5.13a). Given a query semantic label, we score each image $I \in \mathcal{D}$ as follows. We find its k nearest neighbours in $\mathcal{V}$ according to the distance with respect to the CNN descriptor. We set the score of I to the number of neighbours with the same semantic label as the query. Finally, we rank the images in $\mathcal{D}$ according to their score and return them to the user.

To evaluate the system, we manually annotate ground-truth semantic labels on $\mathcal{D}$, and compute the average precision of the five possible queries (fig. 5.14). As baseline, we compare against a system equivalent to the one described above, except that we replace the spatial support used for finding the aspect clusters: CNN on whole frames, rather than on segmentation (sec. 5.5.2). We also compare to an upper bound where we find the aspect clusters using the distance (5.1) between our ground-truth aspect label annotations (sec. 5.3).

Results CNN on segmentation (fig. 5.14, red curve) clearly outperforms CNN on whole frames (blue curve). The only thing changing between the two curves is the method used for aspect discovery: exploiting video to get a segmentation leads to better aspect discovery, which in turn leads to better image retrieval performance. As expected, discovering aspects using the ground-truth annotations provides an upper bound for these automatic methods, showing that further improvements in aspect discovery would be beneficial to tasks like image retrieval (pink curve).

Fig. 5.13 shows a few qualitative examples. Consider the query "Right side" (d): when we use ground-truth labels and CNN on segmentation for clustering, five of the seven highest scoring images match the query, *i.e* the tiger in the retrieved images is actually facing right. This degrades to two when we use CNN on whole frame, showing that in general the aspect discovery system benefits from using the segmentation in this case. Instead, the performance of segmentation and whole frame are very similar on face close-up; in this case the tiger occupies most of the image, which allows CNN on whole frame to match the performance of CNN on segmentation.

5.7.2 Modeling aspect transitions

Another advantage of video over still images is that it allows to reason about transitions across aspects, for example from frontal head to head facing right, or from lying to standing (fig. 5.15). This can be useful in a variety of tasks, such as tracking object instances in new video, aspect-based video retrieval, or as a starting point for learning grammars of aspects.

We consider here learning a probabilistic model of aspect transitions from the ground-truth aspect labels in our video dataset (sec. 5.3). Let $\mathcal{A}$ be the set of all unique aspects in the dataset (for a total of 643, sec. 5.4). We construct a transition matrix T

where each entry

$$T(K,L) = \left(1 - \frac{1}{1+N_k}\right) \cdot P(K,L) + \frac{1}{1+N_k}\Pi(K,L) \quad , \tag{5.2}$$

is the probability of transitioning from aspect K to L. It is computed as the weighted sum of a transition probability P we learn from the ground-truth labels, and a smoothness prior Π (N_k the number of occurrences of aspect K in the dataset).

We compute $P(K,L)$ from the ground-truth aspect labels as follows. Let $(f_i, f_j)^s_{KL}$ be any two frames in a shot s such that f_i contains an instance of aspect K and f_j an instance of aspect L. Each such pair contributes to $P(K,L)$ by $w(i,j) = e^{(1-|j-i|)}$, *i.e* the probability of transitioning from K to L is greater if f_i and f_j are close in time. This gives

$$P(K,L) = \frac{\sum_s \sum_{(f_i,f_j)^s_{KL}} w(i,j)}{Z} \quad , \tag{5.3}$$

where $Z = \sum_{A \in \mathcal{A}} P(K,A)$ is the normalisation constant.

To model the transitions between rare aspects more effectively, we include a smoothness prior Π

$$\Pi(K,L) = \frac{1 - D(K,L)}{\sum_{A \in \mathcal{A}} 1 - D(K,A)} \quad , \tag{5.4}$$

where D is the distance (5.1) between aspects, which is smooth by construction (sec. 5.3).

We demonstrate the expressiveness of the learnt transitions qualitatively, by using T to produce random walks in aspect space (fig. 5.15). We choose the starting aspect A_0 by uniformly sampling from $\mathcal{A}$. At every step t we sample the next aspect A_{t+1} from the transition probability $T(A_{t+1}, A_t)$. To visualize the random walk, for each $A_t = K$ we choose one instance of aspect K from those available in the dataset. This approach discovers several interesting aspect transitions, such as standing up (fig. 5.15, third row): note how the four tigers illustrating this transition all come from different shots.

Figure 5.15: **Aspect transitions.** *We learn transitions between aspects from the labels in our dataset, and use them to generate interesting random walks in aspect space (sec. 5.7.2).* ***Top row:*** *Tiger turning its head.* ***Middle row:*** *Tiger entering the frame and leaving.* ***Last row:*** *Tiger lying down, standing up and walking into tall grass.*

Chapter 6

Conclusions

6.1 Class-agnostic video object segmentation

In chapter 2 we introduced a novel, computationally efficient, unsupervised foreground object segmentation method which was published in (Papazoglou and Ferrari, 2013). Our approach produces a rough initial segmentation of the foreground object relying solely on motion. This initial estimate is then refined by integrating information over the entire video.

6.1.1 Outlook

Our method is generic and computationally efficient, but there are engineering decisions that could be improved. One of the areas that would be simple to improve is to have a better appearance model, such as using a LAB colorspace instead of RGB. Alternatively, more elaborate appearance models could be used such as HOG descriptors (Dalal and Triggs, 2005). However, the biggest disadvantage of the method is its reliance on computing the optical flow in order to produce the initial rough segmentation (sec. 2.3.1). While optical flow estimation can be parallelised on GPUs to lower computation times (Sundaram et al., 2010), it still constitutes the major bottleneck of our method.

The recent work of (Dosovitskiy et al., 2015) has shown that it is possible to estimate optical flow using Convolutional Neural Networks (CNN). This method can achieve a frame rate of 5-10 frames/sec, while still achieving competitive results in standard benchmarks. One could easily imagine substituting the optical flow estimation algorithm of (Sundaram et al., 2010) for (Dosovitskiy et al., 2015) to gain an

immediate speed gain.

However, an interesting approach would be training a CNN to estimate motion boundaries directly, without the need for optical flow. This should be an easier task to train for, as the CNN needs to estimate a binary mask of pixels where motion changes abruptly, rather than estimating the exact motion field itself. Following that line of thought further, we could even imagine training a CNN to produce the rough initial segmentation directly. This would essentially act as jointly training an image and motion saliency estimator. Even if the output segmentation would not be very accurate, our work has shown that a rough initial estimate can be enough to bootstrap a refinement procedure.

Another avenue for speed improvement can be attempting to refine the initial rough segmentation without using an energy minimisation formulation. While graphcuts are generally considered efficient, they can still be an order of magnitude slower compared to geodesic distance transformations (Criminisi et al., 2010). Geodesic distances have been used as an alternative to energy minimisation models for various image processing tasks (Criminisi et al., 2010), including image segmentation, and have been shown to be able to produce comparable results while being an order of magnitude faster. Furthermore, they are naturally parallelisable and can take advantage of modern massively parallel computing architectures on the GPU.

6.2 Class-specific video object segmentation

In chapter 3 we proposed an extension to our unsupervised foreground object segmentation method (chapter 2) for the case of class-specific object segmentation. Our method uses modern object detectors to incorporate prior knowledge aboutincorporates the appearance of the object class, which significantly improves performance compared to the unsupervised setting.

6.2.1 Outlook

We have shown that injecting class-specific knowledge into our segmentation method which can significantly increase performance. To do so, we used a standard object detector (Girshick et al., 2014) as an additional unary potential. What is noteworthy, is that the segmentation algorithm greatly outperforms using just the standalone object detector. Furthermore, the segmentation is dense (all frames are segmented), and typi-

cally better overlaps with the foreground object rather than the detected bounding-box.

A promising avenue for future work would be to try to improve the performance of the object detector by using the output of the segmentation algorithm. While the object detector is able to confidently detect the objects in some frames in the video, it is often the case that it fails to detect any object in subsequent frames. We then could use the segmentation of the frames that the detector fails as additional training data in order to improve its performance. This procedure could even be done iteratively: the improved object detector could lead to better segmentations, which could in turn be used to further improve the object detector.

Another avenue for future work could be multi-class segmentation, where we have multiple interacting objects that we wish to segment. For instance, consider a video of a person playing a violin. In this case we would want to have separate pixel-level segments for the person and the violin. If we used solely an object detector we could have a bounding-box around each object, but as the objects overlap, we would not be able to find the exact spatial extent of each. However, using the detections as input in a video object segmentation framework could allow us to separate them at the pixel-level. A straightforward approach to accomplish that would be to follow a similar energy formulation as in equations 2.6 and 3.1. In the case of multi-class segmentation however, the labels l_i would not be only two (background or foreground) but rather $N+1$, where N is the number of classes in the video. In our example that would correspond to 3 labels (background, person and violin). The location model term $L_i^t(l_i^t)$ is class-agnostic and can only differentiate between foreground and background. As such, it would have the same value for all classes except background. The appearance term $A_i^t(l_i^t)$ can be class-specific by learning an appearance model for each class. Naturally, the class-specific appearance term can be derived from the score of the detector for each class as in sec. 3.3. This model could be efficiently optimised using an algorithm such as alpha-expansion (Delong et al., 2012).

6.3 Temporal alignment of videos based on object viewpoint

In chapter 4 we presented a model and an accompanying optimisation procedure for video temporal alignment of semantically similar scenes, which will be published in the Asian Conference on Computer Vision (ACCV) 2016. Our model is different to

previous works on temporal alignment in that we make no assumption that the videos show the exact same sequence of events, but can cope with unconstrained, realistic videos where events can appear in arbitrary order and multiple times in each video. To evaluate our model, we proposed a new evaluation protocol that is suitable for this setting and also conducted a substantial human study.

6.3.1 Outlook

We presented our model for video temporal alignment. Its strength lies in its ability to align realistic sequences that do not show a scripted series of events. While we focused our experimental analysis on viewpoint similarity, our model is generic: it could be used to align sequences based on any kind of event (*e.g.* pose), provided that an appropriate similarity measure between different frames is used.

Although we believe our model to be a significant improvement for various applications, our approach is not without drawbacks. Below we discuss some of these.

Computational requirements. While our model is more flexible than DTW, performing inference on it is computationally expensive. While our MCMC sampling moves are generally efficient to compute (as most of the computation is reused) we still need to sample for many iterations before we reach a low energy value. In our experiments, we performed 500000 sampling moves for each per pair of videos to be aligned. This takes approximately 13 minutes on an Intel Xeon E3 CPU, clocked at 3.20GHz. On average, we observed that at 10000 samples we already obtain an energy value that is approximately $20-25\%$ higher than the energy after 500000 samples, after which the rate of improvement drops fast. Observing that behaviour, we could dynamically adjust the number of sampling moves so that the inference procedure could stop when the rate of improvement is low.

Training appropriate descriptors. While CNN descriptors are the state-of-the-art for most computer vision tasks nowadays, they typically need to be finetuned for the task at hand. Using CNN descriptors trained on a separate task leads to descriptors that may be invariant to the content that we wish to discriminate (*e.g.* viewpoint). Finetuning the network to be able to differentiate viewpoints can be done following our procedure (sec. 4.5). Finetuning it to differentiate the pose of articulated objects however is not as straightforward and would require expensive manual annotations. While one could use more traditional descriptors in that case (as past works do Rao

et al. (2003); Ukrainitz and Irani (2006); Dexter et al. (2009)), being able to finetune a CNN for this task would be an interesting avenue to explore.

Smoothness across temporal segment changes. Our alignment by construction produces smooth alignments when matching pairs of temporal segments. However, at the point where one segment stops and another begins we may get a brief discontinuity, as the matched segments may come from different parts of the video. While the alignment may be correct in terms of viewpoint, this discontinuity is not visually appealing to humans. One possible solution would be to post-process the segments in order to produce a smoother transition between them (keeping the matches fixed), similar to video morphing. In video morphing, the object in one video is transformed over time into the object in another video. In this case, we would morph the same object from one segment to the next, in order to produce visually smooth transitions between them.

A more principled approach would be to incorporate a smoothness term between subsequent segments in the correspondence likelihood. In that case however, the factors of the correspondence likelihood (eq. 4.2) would not be conditionally independent anymore. This would produce a more complex model and would make inference even more computationally expensive.

6.4 Discovering object aspects from video

In chapter 5 we presented an extensive exploration of weakly-supervised aspect discovery from video, which we posed as an image clustering problem. We experimented with several modern appearance descriptors and carefully evaluated the benefits of exploiting video over still images for the task. Furthermore, we assembled a large video dataset for evaluation and proposed a novel protocol to evaluate aspect discovery directly.

6.4.1 Outlook

We experimented with several modern appearance descriptors (SIFT, shape contexts, CNN features), and various levels of spatial support (e.g. whole image, segmentation). We demonstrated that exploiting the nature of video through the use of automatic foreground segmentation leads to consistently better aspect discovery in all cases. Finally, we showed that aspect discovery can enable new applications, such as semantic-aspect

image retrieval, and modelling transitions between aspects.

An avenue for future improvement would be in increasing the discriminative power of the CNN features in regards to pose and viewpoint. Finetuning the network to differentiate different viewpoints as in chapter 4 would most likely bring a significant improvement to the compactness of the discovered aspects. This setting however, would require more supervision as it would need ground-truth annotations for different viewpoints. A possible way to make CNN features more discriminative with regards to viewpoint and pose, without using ground-truth annotations, would be an unsupervised visual representation learning approach such as the one presented in (Doersch et al., 2015).

Further improvements could be had with regard to the aspect discovery method. In this chapter we exploited the temporal nature of video to localise the objects through video object segmentation. This, however, does not take full advantage of video. For instance, we have observed that aspects change smoothly over time and that the aspect changes within a shot are relatively few (less than three). This information indicates that there should be a force promoting aspect label smoothness for frames close in time. Furthermore, if two frames in different videos are given the same aspect label, other frames that are close to them in time should also have a similar aspect label. This of course cannot be captured by a simple clustering algorithm which considers frames to be completely independent.

A natural way to model such interactions would be an alternating optimisation approach. First, we would learn an appearance model of each aspect label (*e.g.* a GMM or even a simple medoid of the frames having that aspect). This model can be initialised from the discovered aspects of a simple clustering technique as shown in this chapter. We could then refine the aspect label assignments to promote temporal smoothness using an energy formulation with unary and pairwise terms. Thinking of the energy formulation as a graph, each frame would correspond to a different node. Nodes (frames) of the same video that are subsequent in time would be connected, while nodes that belong to different videos or are not subsequent would not be connected. The pairwise terms in our energy formulation would be zero if the corresponding pair of nodes has the same aspect label and non-zero otherwise, which would promote subsequent frames having the same aspect label. The unary terms would correspond to the likelihood of a node having a specific aspect. This energy formulation could be efficiently optimised using an algorithm such as alpha-expansion (Delong et al., 2012). Having refined the aspect label assignments, the appearance model could be updated and the

procedure would be repeated until convergence.

Appendix A

User study evaluation protocol

Here we describe the instructions given to the participants of the user study. The participants were first given the instructions displayed on fig. A.1. Then they were presented with a video and to possible alignments and were asked to pick the best one as showin in fig. A.2. As shown in the images, the participants were never told which alignment belongs to which video. In order to ensure the quality of evaluation, we also gave the participants some test comparisons among the evaluations, which compared a ground-truth alignment versus a random alignment. Participants that failed these test questions were removed from the study, and their answers were excluded from the results reported in sec. 4.6.4.

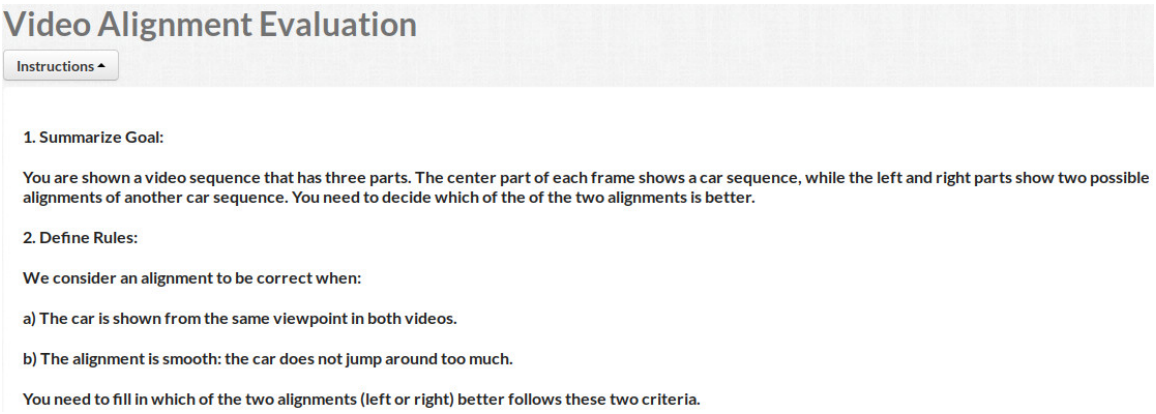

Figure A.1: The instructions given to participants of the user study. The users were asked to select the better alignment method in terms of viewpoint correctness and visual pleasingness (smooth transitions without stuttering).

Which of the two video better aligns to the center one:

- The left one
- The right one

Which of the two video better aligns to the center one:

- The left one
- The right one

Figure A.2: The instructions given to participants of the user study. The users were asked to select the better alignment method in terms of viewpoint correctness and visual pleasingness (smooth transitions without stuttering).

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on PAMI*, 34(11):2274–2282.

Agarwala, A., Zheng, K. C., Pal, C., Agrawala, M., Cohen, M., Curless, B., and Szeliski, R. (2005). Panoramic video textures. In *SIGGRAPH*.

Aghazadeh, O., Azizpour, H., Sullivan, J., and Carlsson, S. (2012). Mixture component identification and learning for visual recognition. In *ECCV*.

Akin, O. and Mikolajczyk, K. (2014). Online learning and detection with part-based, circulant structure. In *Proc. ICPR*.

Andriyenko, A. and Schindler, K. (2011). Multi-target tracking by continuous energy minimisation. In *CVPR*.

Azizpour, H. and Laptev, I. (2012). Object detection using strongly-supervised deformable part models. In *ECCV*.

Bai, X., Wang, J., Simons, D., and Sapiro, G. (2009). Video snapcut: robust video object cutout using localized classifiers. In *SIGGRAPH*.

Barnichm, O. and Van Droogenbroeck, M. (2011). Vibe: A universal background subtraction algorithm for video sequences. *IEEE Trans. Image Processing*.

Belongie, S. and Malik, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. on PAMI*, 24(24).

Belongie, S., Malik, J., and Puzicha, J. (2002). Matching with shape contexts. *IEEE Trans. on PAMI*, 24(4):509–522.

Bishop, C. (2006). Pattern recognition and machine learning. *Springer*.

Blank, M., Gorelick, L., Shechtman, E., Irani, M., and Basri, R. (2005). Actions as space-time shapes. In *ICCV*.

Bourdev, L., Maji, S., Brox, T., and Malik, J. (2010). Detecting people using mutually consistent poselet activations. In *ECCV*.

Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*.

Bowyer, K., Stewman, J., Stark, L., and Eggert, D. (1988). Errors-2: a 3d object recognition system using aspect graphs. In *Proc. ICPR*.

Brau, E., J., G., Simek, K., Del Pero, L., Dawson, C. R., and Barnard, K. (2013). Bayesian 3d tracking from monocular video. In *ICCV*.

Brox, T., Bourdev, L., Maji, S., and Malik, J. (2011). Object segmentation by alignment of poselet activations to image contours. In *CVPR*, pages 2225–2232.

Brox, T. and Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *ECCV*.

Brutzer, S., Hoeferlin, B., and Heidemann, G. (2011). Evaluation of background subtraction techniques for video surveillance. In *CVPR*.

C., F. (1984). Summed-area tables for texture mapping. In *SIGGRAPH*.

Carreira, J. and Sminchisescu, C. (2010). Constrained parametric min-cuts for automatic object segmentation. In *CVPR*.

Caspi, Y. and Irani, M. (2000). A step towards sequence-to-sequence alignment. In *CVPR*.

Caspi, Y. and Irani, M. (2001). Alignment of non-overlapping sequences. In *ECCV*.

Caspi, Y. and Irani, M. (2002). Spatio-temporal alignment of sequences. *IEEE Trans. on PAMI*.

Caspi, Y., Simakov, D., and Irani, M. (2006). Feature-based sequence-to-sequence matching. *IJCV*, 68(1):53–64.

Chang, J., Wei, D., and Fisher III, J. W. (2013). A video representation using temporal superpixels. In *CVPR*.

Chockalingam, P., Pradeep, S. N., and Birchfield, S. (2009). Adaptive fragments-based tracking of non-rigid objects using level sets. In *ICCV*.

Criminisi, A., Sharp, T., Rother, C., and Perez, P. (2010). Geodesic image and video editing. In *ACM Transactions on Graphics*.

Criminisi, A., Shotton, J., and Konukoglu, E. (2011). Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*.

Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2003). Detecting moving objects, ghosts, and shadows in video streams. *IEEE Trans. on PAMI*.

Cyr, C. M. and Kimia, B. B. (2001). 3d object recognition using shape similiarity-based aspect graph. In *ICCV*.

Dalal, N. and Triggs, B. (2005). Histogram of Oriented Gradients for human detection. In *CVPR*.

Delong, A., Osokin, A., Isack, H. N., and Boykov, Y. (2012). Fast approximate energy minimization with label costs. *IJCV*.

Dexter, E., Perez, P., and Laptev, I. (2009). Multi-view synchronization of human actions and dynamic scenes. In *BMVC*.

Diego, F., Serrat, J., and Lpez, A. M. (2013). Joint spatio-temporal alignment of sequences. In *IEEE Transactions on Multimedia*.

Divvala, S., Efros, A., and Hebert, M. (2012). How important are 'deformable parts' in the deformable parts model? In *ECCV*.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *ICCV*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*.

Dong, J., Xia, W., Chen, Q., Feng, J., Huang, Z., and Yan, S. (2013). Subcategory-aware object classification. In *CVPR*.

Dosovitskiy, A., Fischer, P., Ilg, E., Husser, P., Hazrba, C., Golkov, V., Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *ICCV*.

Douze, M., Revaud, J., Verbeek, J., Jegou, H., and Schmid, C. (2016). Circulant temporal encoding for video retrieval and temporal alignment. *IJCV*.

Drayer, B. and Brox, T. (2014). Training deformable object models for human detection based on alignment and clustering. In *ECCV*.

Endres, I. and Hoiem, D. (2010). Category independent object proposals. In *ECCV*.

Evangelidis, G. D. and Bauckhage, C. (2013). Efficient subframe video alignment using short descriptors. *IEEE Trans. on PAMI*.

Everingham, M. et al. (2010). The PASCAL Visual Object Classes Challenge 2010 Results.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

F. L. C. Padua, R. L. C. (2009). Linear sequence-to-sequence alignment. *IEEE Trans. on PAMI*.

Faktor, A. and Irani, M. (2014). Video object segmentation by non-local consensus voting. In *BMVC*.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9).

Ferrari, V., Tuytelaars, T., and Van Gool, L. (2001). Real-time affine region tracking and coplanar grouping. In *CVPR*.

Foley, J., van Dam, A., Feiner, S. K., and Hughes, J. (1990). *Computer Graphics: Principles and Practice, 2nd Edition*. Addison-Wesley.

Giordano, D., Murabito, F., Palazzo, S., and Spampinato, C. (2015). Superpixel-based video object segmentation using perceptual organization and location prior. *CVPR*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010). Efficient hierarchical graph-based video segmentation. In *CVPR*.

Gu, C., Arbeláez, P., Lin, Y., Yu, K., and Malik, J. (2012). Multi-component models for object detection. In *ECCV*.

Gu, C. and Ren, X. (2010). Discriminative mixture-of-templates for viewpoint classification. In *ECCV*.

Huang, C. and Nevatia, R. (2010). High performance object detection by collaborative learning of joint ranking of granule features. In *CVPR*.

Jain, A., Chatterjee, S., and Vidal, R. (2013). Coarse-to-fine semantic video segmentation using supervoxel trees. In *ICCV*.

Jain, S. D. and Grauman, K. (2014). Supervoxel-consistent foreground propagation in video. In *ECCV*.

Jia, Y. (2013). Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`.

Jiang, Y., Ngo, C., and Yang, J. (2007). Towards optimal bag-of-features for object categorization and semantic video retrieval. In *International Conference on Image and Video retrieval*.

Joulin, A., Tang, K., and Fei-Fei, L. (2014). Efficient image and video co-localization with frank-wolfe algorithm. In *ECCV*.

Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R. (2007). High dynamic range video. *ACM Transactions on Graphics*.

Kim, W.-H. and Kim, J.-N. (2009). An adaptive shot change detection algorithm using an average of absolute difference histogram within extension sliding window. In *ISCE*.

Koenderink, J. J. and van Doorn, A. J. (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics*.

Kong, Y., Jia, Y., and Fu, Y. (2013). Ochs, p. and malik, j. and brox, t. *IEEE Trans. on PAMI*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kuo, C., Huang, C., and Nevatia, R. (2009). Multi-target tracking by on-line learned discriminative appearance models. In *ICCV*.

Kwak, S., Cho, M. Laptev, I., Ponce, J., and Schmidt, C. (2015). Unsupervised object discovery and tracking in video collections. In *ICCV*.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*.

Lee, Y. J., Ghosh, J., and Grauman, K. (2012). Discovering important people and objects for egocentric video summarization. In *CVPR*.

Lee, Y. J., Kim, J., and Grauman, K. (2011). Key-segments for video object segmentation. In *ICCV*.

Li, F., Kim, T., Humayun, A., Tsai, D., and Rehg, J. M. (2013). Video segmentation by tracking many figure-ground segments. In *ICCV*.

Li, Y., Huang, C., and Nevatia, R. (2009). Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*.

Liao, J., Lima, R. S., Nehab, D., Hoppe, H., and Sander, P. V. (2014). Semi-automated video morphing. In *Eurographics Symposium on Rendering*.

Liebelt, J. and Schmid, C. (2010). Multi-view object class detection with a 3d geometric model. In *CVPR*.

Liebelt, J., Schmid, C., and Schertler, K. (2008). Viewpoint-independent object class detection using 3d feature maps. In *CVPR*.

Liu, C., Freeman, W. T., Adelson, E. H., , and Weiss, Y. (2008). Human-assisted motion annotation. In *CVPR*.

Liu, X., Tao, D., Song, M., Ruan, Y., Chen, C., and Bu, J. (2014). Weakly supervised multiclass video segmentation. In *CVPR*.

Lowe, D. (2001). Local feature view clustering for 3D object recognition. In *CVPR*, pages 682–688. Springer.

Ma, T. and Latecki, L. J. (2012). Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*.

Malisiewicz, T. (2011). Ensemble of exemplar-svms, implementation. https://github.com/quantombone/exemplarsvm.

Manning, C. D., Raghavan, P., and Schtze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Marin-Jimenez, M. J., Zisserman, A., Eichner, M., and Ferrari, V. (2014). Detecting people looking at each other in videos. *IJCV*.

Mei, L., Liu, J., Hero, A., and Savarese, S. (2011). Robust object pose estimation via statistical manifold modeling. In *ICCV*.

Nagaraja, N. S., Schmidt, F. R., and Brox, T. (2015). Video segmentation with just a few strokes. In *ICCV*.

Neal, R. M. (1993). Probabilistic inference using markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

Ngo, C., Ma, Y., and Zhang, H. (2005). Video summarization and scene detection by graph modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2).

Ochs, P. and Brox, T. (2012). Higher order motion models and spectral clustering. In *CVPR*.

Oh, S., Russell, S. J., and Sastry, S. (2009). Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 54(3):481–497.

Oneata, D., Revaud, J., Verbeek, J., and Schmid, C. (2014). Spatio-temporal object detection proposals. In *ECCV*.

Palazzo, S., Spampinato, C., and Giordano, D. (2016). Gamifying video object segmentation. *arXiv*, arXiv:1601.00825v1.

Papazoglou, A., Del Pero, L., and Ferrari, V. (2016). Discovering object aspects from video. *Image and Vision Computing*, 52(8):206–217.

Papazoglou, A. and Ferrari, V. (2013). Fast object segmentation in unconstrained video. In *ICCV*.

Perazzi, F., Krhenbl, P., Pritch, Y., and Hornung, A. (2012). Saliency filters: Contrast based filtering for salient region detection.

Perez-Rua, J., Crivelli, T., and Prez, P. (2015). Background-foreground tracking for video object segmentation. *Proceedings of the IEEE International Conference on Image Processing*.

Plantinga, W. H. and Dyer, C. R. (1986). An algorithm for constructing the aspect graph. In *FOCS*.

Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. (2012). Learning object class detectors from weakly annotated video. In *CVPR*.

Price, B. L., Morse, B. S., and Cohen, S. (2009). Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*.

Puscas, M. M., Sangineto, E., Culibrk, D., and Sebe, N. (2015). Unsupervised tube extraction using transductive learning and dense trajectories. In *ICCV*.

Rao, C., Gritai, A., and Shah, M. (2003). View-invariant alignment and matching of video sequences. In *ICCV*.

Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop at CVPR*.

Rochan, M. and Wang, Y. (2014). Efficient object localisation and segmentation in weakly labeled videos. *Advances in Visual Computing*.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*.

Ruegg, J., Wang, O., Smolic, A., and Gross, M. (2013). Ducttake: Spatiotemporal video compositing. *Comput. Graphics Forum (Proc. Eurographics)*, 32(2).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *IJCV*.

Sakoe, H. and Chiba, S. (1978). Object segmentation by alignment of poselet activations to image contours. In *IEEE Trans. Acoustics, Speech, and Signal Proc.*

Santhoshkumar, S., Karthikeyan, S., and Manjunath, B. S. (2013). Robust multiple object tracking by detection with interacting markov chain monte carlo. In *Proceedings of the IEEE International Conference on Image Processing*.

Savarese, S. and Fei-Fei, L. (2008). View synthesis for recognizing unseen poses of object classes. In *ECCV*.

Sharif, M. D. H., Martinet, J., and Djeraba, C. (2008). *Motion Saliency*. Springer US.

Spampinato, C., Palazzo, S., Murabito, F., and Giordano, D. (2015). Using the eyes to "see" the objects. In *ACM Multimedia*.

Stretcu, O. and Leordeanu, M. (2015). Multiple frames matching for object discovery in video. In *BMVC*.

Su, H., Sun, M., Fei-Fei, L., and Savarese, S. (2009). Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV*.

Sundaram, N., Brox, T., and Keutzer, K. (2010). Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*.

Sundberg, P., Brox, T., Maire, M., Arbelaez, P., and Malik, J. (2011). Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*.

Tang, K., Sukthankar, R., Yagnik, J., and Fei-Fei, L. (2013). Discriminative segment annotation in weakly labeled video. In *CVPR*.

Thomas, A., Ferrari, V., Leibe, B., Tuytelaars, T., Schiele, B., and Van Gool, L. (2006). Towards multi-view object class detection. In *CVPR*.

Trichet, R. and Nevatia, R. (2013). Video segmentation with spatio-temporal tubes. In *International Conference on Advanced Video and Signal Based Surveillance*.

Tsai, D., Flagg, M., and Rehg, J. (2010). Motion coherent tracking with multi-label mrf optimization. In *BMVC*.

Tuytelaars, T. and van Gool, L. (2004). Synchronizing video sequences. In *CVPR*.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *IJCV*.

Ukrainitz, Y. and Irani, M. (2006). Aligning sequences and actions by maximizing space-time correlations. In *ECCV*.

Viola, P. and Jones, M. (2001). Robust real-time object detection. In *IJCV*, volume 1.

Wang, H. and Wang, T. (2016). Primary object discovery and segmentation in videos via graph-based transductive inference. *CVIU*.

Wang, L., Hua, G., Sukthankar, R., Xue, J., and Zheng, N. (2014a). Video object discovery and co-segmentation with extremely weak supervision. In *ECCV*.

Wang, O., Schroers, C., Zimmer, H., Gross, M., and Sorkine-Hornung, A. (2014b). Videosnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics*.

Wang, T. and Collomosse, J. (2012). Probabilistic motion diffusion of labeling priors for coherent video segmentation. *IEEE Transactions on Image Processing*.

Wang, T. and Wang, H. (2014). Graph transduction learning of object proposals for video object segmentation.

Wang, W., Shen, J., and Porikli, F. (2015). Semantic object segmentation via detection in weakly labeled video.

Wen, L., Du, D., Lei, Z., Li, S. Z., and Yang, M. (2015). Jots: Joint online tracking and segmentation. *CVPR*.

Wolf., L. and Zomet, A. (2006). Wide baseline matching between unsynchronized video sequences. *IJCV*.

Wu, Z., Li, F., Sukthankar, R., and Rehg, J. M. (2015). Robust video segment proposals with painless occlusion handling. In *CVPR*.

Yang, B. and Nevatia, R. (2014). Multi-target tracking by online learning a crf model of appearance and motion patterns. *IJCV*.

Yang, J., Price, B., Shen, X., Lin, Z., and Yuan, J. (2016). Fast appearance modelling for automatic primary video object segmentation. *Transactions on Image Processing*.

Zhang, D., J. O. and Shah, M. (2013). Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*.

Zhang, N., Farrell, R., and Darrell, T. (2012). Pose pooling kernels for sub-category recognition. In *CVPR*.

Zhang, Y., Chen, X., Li, J., Wang, C., and Xia, C. (2015). Semantic object segmentation via detection in weakly labeled video.